软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(7):2183–2203 [doi: 10.13328/j.cnki.jos.006263]

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

针对复杂用户评论的代码质量属性判断*

徐海燕[1,2], 姜 瑛[1,2]

[1](云南省计算机技术应用重点实验室,云南 昆明 650504)

[2](昆明理工大学 信息工程与自动化学院,云南 昆明 650504)

通讯作者: 姜瑛, E-mail: jy_910@163.com

摘 要: 随着开发者社区和代码托管平台成为程序员获取代码的主要途径,针对代码的用户评论数量急剧增加.用户在使用代码后给出的评论中包含多种静态和动态的代码质量属性信息,但是由于用户评论多为复杂句,使得评论中包含的代码质量属性难以判断.针对复杂用户评论的代码质量属性判断将有助于分析用户评论中的代码质量信息,有助于开发者在了解用户的代码使用情况和用户关注的代码质量属性后有针对性地提升代码质量.提出了针对复杂用户评论的代码质量属性判断方法.首先对复杂用户评论进行分句并构建分句的依存句法关系有向图;然后,应用基于分句的依存句法关系的主题判断规则抽取分句中的主题;接着,根据初始的代码质量属性特征词库识别各主题对应的代码质量属性,并获取各主题的代码质量属性表现与表现结果;最后,基于主题处理规则分析复杂用户评论中的代码质量属性表现与表现结果,产生复杂用户评论中代码质量属性相关结果,并持续扩充初始代码质量属性特征词库.实验结果表明,该方法能够对复杂用户评论的代码质量属性进行有效判断.

关键词: 复杂用户评论;代码质量属性;主题判断规则;代码质量属性表现;代码质量属性表现结果;主题处理规则

中图法分类号: TP311

中文引用格式: 徐海燕,姜瑛.针对复杂用户评论的代码质量属性判断.软件学报,2021,32(7):2183–2203. http://www.jos.org.cn/1000-9825/6263.htm

英文引用格式: Xu HY, Jiang Y. Determination of code quality attribute for complex user's comments. Ruan Jian Xue Bao/Journal of Software, 2021,32(7):2183–2203 (in Chinese). http://www.jos.org.cn/1000-9825/6263.htm

Determination of Code Quality Attribute for Complex User's Comments

XU Hai-Yan[1,2], JIANG Ying[1,2]

[1](Yunnan Key Laboratory of Computer Technology Application, Kunming 650504, China)

[2](Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650504, China)

Abstract: As the developer community and code-hosting platforms become the primary means for programmers to access code, the number of user's comments on code has increased dramatically. There are a variety of static and dynamic code quality attributes in user's comments. However, as most of the user's comments are complex sentences, it is difficult to identify the code quality attributes in the comments. Judging the code quality attributes of complex user's comments will help to analyze the code quality information in user's comments and to improve code quality for the developers when they know about user's code usage and code quality attributes. In this study, a method is proposed to judge code quality attributes based on complex user's comments. Firstly, complex user's comments are

* 基金项目: 国家重点研发计划(2018YFB1003904); 国家自然科学基金(61462049, 61063006, 60703116); 云南省应用基础研究计划(2017FA033); 云南省计算机技术应用重点实验室开放基金(2020101)

Foundation item: National Key R&D Program of China (2018YFB1003904); National Natural Science Foundation of China (61462049, 60703116, 61063006); Key Project of Yunnan Applied Basic Research (2017FA033); Open Foundation of Yunnan Key Laboratory of Computer Technology Application (2020101)

本文由"面向非确定性的软件质量保障方法与技术"专题特约编辑陈俊洁副教授、汤恩义副教授、何啸副教授以及马晓星教授推荐.

收稿时间: 2020-09-14; 修改时间: 2020-10-26; 采用时间: 2020-12-14; jos 在线出版时间: 2021-01-22

divided into clauses and a dependency syntactic relation directed graph of the clauses is constructed. After that, the topic of the clause is extracted based on the topic judgment rule of the dependency syntactic relation of the clause. Then, according to the initial feature thesaurus of code quality attribute, the code quality attributes corresponding to each topic are identified, and the representation and the representation result of code quality attribute for each topic are acquired. And finally, the representation and the representation result of code quality attribute in the complex user's comments are analyzed based on the topic processing rule. The code quality attribute related result in the complex user's comment is produced, and the initial code quality attribute feature thesaurus is continuously expanded. The experimental results show that the proposed method can judge the code quality attributes of complex user's comments effectively.

Key words: complex user's comments; code quality attribute; topic judgment rule; representation of code quality attribute; representation result of code quality attribute; topic processing rule

1 引 言

代码质量研究对软件开发具有重要意义.当前,代码质量的度量被广泛应用于软件工程的不同领域,用以预测软件缺陷、软件质量、安全漏洞、可维护性等.Berkhan 等人[1]对代码进行静态代码分析从而度量软件组件质量.Mamun 等人[2]通过对软件与代码质量的相关性进行实证研究,发现代码质量可以指示软件的可维护性,而且预测这样的代码质量可以为软件在未来的版本中提供技术债务的指示.Kaur 等人[3]通过静态代码分析过程进行漏洞检测,利用自动静态代码分析工具对软件开发过程中的源代码质量进行审计,以识别软件潜在的安全漏洞.Ruiz-Rube 等人[4]认为,提高软件产品的质量可以提高客户满意度并提供竞争优势,静态程序分析是验证软件质量特性的一种有效技术,使用自动工具分析源代码可以发现不同的代码味道和潜在的错误.近年来,随着代码开源和代码复用的发展,大量开发人员聚集在 Github、开源中国、CSDN 等代码托管平台和开发者社区上,开发人员上传的大量代码已成为开发者编程的重要参考.但是,由于代码开发者的编程水平和编程经验参差不齐,代码的质量差别较大,用户在使用代码时会出现各种问题.在软件产品中,Panago 等人[5]认为,因为软件开发者和软件使用者并不认识,软件使用者的使用体验反馈,如建议意见等,对开发者十分重要.Xu 等人[6]认为,类似于软件产品,代码作为一种特殊的产品,其使用者在使用时会针对代码的实际使用情况给出相应的评论,可以通过用户评论建立开发者与用户之间的联系.因此,通过分析针对代码的用户评论中的代码质量属性相关信息,能够为代码开发者提供用户的代码使用反馈信息,帮助开发者提高代码质量,并为新用户选择代码提供参考.

在汉语中,复杂句式主要是指有两个以上互不从属的分句通过特定连词连接而成,句子中有较小停顿,句子前后有较大停顿[7].与普通的产品用户评论不同,针对代码的用户评论多为复杂句,且具有评论文本复杂度高、评论领域性强、评论粒度差别大、文本长短不一、不规则的特点[6].此外,复杂用户评论中一般包含多种代码质量属性,且包含的代码质量属性之间具有多种逻辑关系,导致判断用户评论中代码质量属性信息困难.因此,如何针对复杂的代码用户评论进行分析,并从中获取与代码质量属性相关的信息成为亟需解决的问题.

本文的主要贡献如下:

(1) 基于复杂用户评论分句的依存句法关系,制定了基于依存句法关系建立的主题判断规则,对复杂用户评论分句中存在的主题进行了抽取;然后,应用代码质量属性特征词库识别出各主题对应的代码质量属性.本文采用先抽取复杂用户评论主题再进行主题对应代码质量属性识别的方法,可以对复杂用户评论中可能存在的代码质量属性进行较为完整的判断.

(2) 对于复杂用户评论进行主题抽取后,基于主题间的关系,制定了同一主题处理规则,对复杂用户评论中同一代码质量属性对应的多个相同主题进行了合并,针对不同主题也进行了相关处理.通过主题处理规则,可以对抽取出的复杂用户评论中的主题进行处理,从而综合判断出复杂用户评论的代码质量属性.最后,根据主题处理规则获取的复杂用户评论的代码质量属性表现与表现结果,构建新的特征词对,扩充了初始的代码质量属性特征词库,并通过特征词库的动态扩充,不断提升复杂用户评论中代码质量属性识别的精度.

本文第 2 节介绍用户评论与复杂句的相关研究工作.第 3 节对静态和动态代码质量属性定义以及代码质量属性特征词库进行说明.第 4 节介绍针对复杂用户评论的代码质量属性判断方法.第 5 节在真实数据上进行实验.最后总结全文.

2 相关工作

随着互联网的发展,用户习惯通过在线网络平台发表对产品的看法,在线用户评论成为用户了解产品质量等相关信息的一个快速、有效的途径.Hu 等人[8]针对 APP 软件的用户评论,应用评论种子循环挖掘出用户评论中体现不同类型的 APP 软件使用反馈的用户评论.Duan 等人[9]通过定义 APP 软件缺陷抽取规则,利用缺陷句型结构挖掘 APP 软件用户反馈中的软件缺陷,并对软件的缺陷程度进行了分析.代码与 APP 软件产品相似,作为一种特殊的产品,随着代码托管平台和开发者社区成为开发人员获取代码的主要途径,代码使用者会根据代码的使用情况对代码进行评论,针对代码的用户评论中包含了丰富的代码质量信息.Xu 等人[6]基于用户评论,根据评价对象和评价句型的识别规则,对代码质量进行识别和分析.然而,在针对软件产品的用户评论和针对代码的用户评论中,用户评论不仅仅是简单句,尤其在针对代码的用户评论中,用户评论多为复杂句.相较于简单用户评论,复杂用户评论包含了更为丰富和复杂的代码质量属性信息.若是按照一般基于句子级别的规则,虽然能够识别出一些用户评论中的有价值信息,但是识别出的信息较为有限,甚至可能会丢失信息原有的关联性.在部分针对软件质量和代码质量判断的相关工作中,Wang 等人[10]基于 ISO/IEC 9126 软件质量模型,以可用性、可靠性、安全性、实时性、可维护性和可生存性构成了软件可信属性模型.Venkatasubramanyam 等人[11]使用基于 ISO/IEC 9126 质量模型定义的 EMISQ(evaluation method for internal software quality)模型进行静态代码质量分析,使用 DASIQ(dynamic analysis for internal software quality)模型进行动态代码质量分析,并将静态和动态分析获取的代码质量结果进行详细对比.在这些相关工作中,主要针对基于 ISO/IEC 9126 模型的软件质量和代码质量的部分质量属性进行判断.

在部分针对复杂用户评论的研究中,Liu 等人[12]对于嵌套句型等较为复杂的句型结构,在句法分析过程中实施动作层次分解,将复杂语句分解为简单的基本句型.此外,还提出对倒装句型的识别机制,通过匹配接近的句型进行实体移位,调整语序,以便于挖掘文本中实体-动作关联关系.文献[13]建立了一个适用于长句子和复杂句汉语句法分析的论元关系模型,使用多值递归函数识别句型结构并划分句法功能语块,根据句型结构抽取动作的施体和受体,建立论元关系模型.Li 等人[7]在针对篇章级的文本情感倾向的判断中,针对复杂句的情感分析,按照复杂句的转折句式、条件句式等进行基于句式结构的情感极性判断,对于简单句的情感分析则基于朴素贝叶斯进行判断.Mao 等人[14]融合了从底向上和自顶向下的中文树库的复杂句标注方法,将复杂句切分成结构较简单的块进行分析.Ye 等人[15]针对现有问答系统中结构复杂的问句进行研究,首先基于条件随机场算法对问句进行分类和主体识别,然后结合谓词词典和句法分析识别出问句的谓词,最后通过谓词消歧方法来解决相同问句具有不同表达方式的问题.Swain[16]等人针对科技相关的复杂句子和单词,先通过复杂词识别,再通过简单的同义词替换复杂词,对复杂句进行了文本简化.Siddharthan 等人[17]描述了在对复杂句按照句法简化过程中如何解决句子排序、词的选择、指称表达的生成、决定词的选择和代名词的使用等各种生成问题,以保持连接和回指的衔接关系.

通过分析上述研究,我们发现:

1) 在当前针对用户评论的大部分研究中,对简单用户评论和复杂用户评论采取同一种处理方法提取用户评论中的有价值信息.例如,文献[6,8,9]中,对于用户评论多是通过词性规则或句型规则来识别评论中包含的有价值信息.由于简单用户评论与复杂用户评论包含的有价值的信息量存在差异,若是按照一般基于句子级别的规则对复杂用户评论进行处理,那么,当复杂用户评论中存在多个代码质量属性时,虽然能够判断出一些质量属性,但是判断出的质量属性较为有限.此外,文献[10]中对软件质量仅基于可靠性、可用性等 6 个质量属性进行了判断;文献[11]对代码质量只是基于静态分析或者动态测试模型判断少数几个质量属性,质量属性的判断结果较为片面.

2) 部分研究对复杂句的处理主要是针对复杂句中的复杂句式结构或成分进行简化,再对简化后的复杂句进行相应的处理,如文献[12,13]将复杂句式分解为简单句,文献[16]使用简单词替换复杂句中的复杂词.然而,复杂用户评论中通常包含多个代码质量属性,且多个代码质量属性之间存在一定的逻辑关系.因此,若是按照简化复杂句式结构或成分的方法处理复杂用户评论,多个代码质量属性之间的关系并未被简化和处理,将难以判断

提取的代码质量属性是否准确.

因此,本文针对复杂用户评论,基于前期工作提出的11个静态和动态代码质量属性进行主题判断处理,并对复杂用户评论中包含的代码质量属性之间的逻辑关系进行分析,提出了一种针对复杂用户评论的代码质量属性判断方法.

3 代码质量属性

为了判断复杂用户评论中的代码质量属性,对代码质量属性的定义十分重要.代码不经过编译运行就具有的质量属性一般作为代码的静态质量属性,例如代码规模、代码命名、代码结构、编程规范、控制流逻辑等.代码在编译运行时体现出的行为一般与代码的动态质量相关联,例如代码是否能够进行编译、代码的运行速度以及内存占用、代码是否存在错误等.在针对代码的用户评论中,由于不同的开发人员对代码质量的要求存在差别,用户在使用代码时也会评判代码质量,因此用户评论中不仅包含代码的静态质量属性信息,还包括大量代码在测试编译运行过程中才能体现出来的动态代码质量属性信息.

结合代码静态分析和代码动态测试的相关研究[18−20],参考McCall软件质量度量模型[21],本文在前期工作[6]中针对代码静态质量和动态质量分别划分出了4个静态代码质量属性和7个动态代码质量属性,各个质量属性之间互不重叠.

此外,针对代码的用户评论具有的强领域性特点,代码质量属性可以通过评论中的大量领域性词语来进行标识.我们在前期工作[6]中发现,名词通常与代码静态质量具有强相关性,动词一般与代码的动态质量具有强相关性.基于这种规律,针对 11 个代码质量属性,从大量用户评论中人工抽取了具有强相关性及代表性的特征词,例如“缩进”与编程规范性相关,“编译”则与可测试性相关.然而,应用单个特征词虽然可以识别出复杂用户评论中的代码质量属性,但是识别的准确度不高.通过进一步对复杂用户评论进行分析,当单个特征词与不同词组合时,可以更加准确地标识出不同的代码质量属性.例如“版本更新”与“版本不兼容”中都包含了“版本”,但是分别表示了代码的可维护性与可移植性;“运行不成功”与“运行出错”中都包含了“运行”,但是分别表示了代码的可测试性与可靠性.为了能够准确判断出代码质量属性,本文将这种由多个特征词组合并可以准确描述某种代码质量属性的词对定义为特征词对.本文通过分析大量针对代码的用户评论,针对不同代码质量属性具有的不同属性特征,将特征词扩展为特征词对,并在文献[6]给出的代码质量属性特征词的基础上,扩充了前期构建的代码质量属性特征词库,形成了新的初始代码质量属性特征词库.

综上,结合相关研究和用户评论,本文对11个代码质量属性进行了定义,并给出了每个代码质量属性的特征词对示例,见表1.

Table 1 Definition of code quality attributes

表 1 代码质量属性的定义

类型	质量属性	质量属性定义	特征词对示例
静态质量	可读性	代码以简单易理解的方式实现其功能,并容易为用户所分析的性质	代码 臃肿\看 不 明白\写 清楚
	编程规范性	代码符合编程语言的命名和结构规范,以及设计开发中的规范性要求	格式 规范\变量 定义\函数 命名
	可重用性	代码所提供的功能具有应用范围广、代码之间依赖性小、代码内依赖性大的性质	代码 完整\代码 不全\不 受用
	逻辑性	代码的算术计算、控制流程和设计开发逻辑运行正确	数据 溢出\数组 越界\控制流 设计
动态质量	功能性	代码实现满足用户的功能,且功能全部或部分有效可用	用户 权限\登录 不 可用\验证 不 成功
	可测试性	代码在给定的测试环境下,可支持编译测试的程度	编译 成功\执行 失败\搭建 不 起来
	可维护性	代码在新的功能改善和扩充时,可被修改的容易程度	路径 失效\代码 升级\版本 更新
	运行性能	代码在其运行过程中表现的性能,包括时间复杂度、空间复杂度和实现效果	速度 慢\内存 占用 多\显示 不 出来\加载 不了
	可靠性	代码在其他环境或操作下能否保证运行正确的性质	运行 报错\有 问题\程序 崩溃
	可移植性	在代码运行条件发生改变时无需多做修改就可运行,即一次编辑到处运行的性质	版本 不 兼容\环境 不 支持\代码 集成
	易用性	代码简洁明了,具有详细的文档使用说明,用户易于操作	使用 说明\上手 简单\怎么 使用

表1中给出了代码的静态和动态质量属性的划分和定义以及每个质量属性的特征词对示例,将为第4节针对复杂用户评论的代码质量属性判断提供依据.

4 针对复杂用户评论的代码质量属性判断

由于代码的质量属性信息只有在用户使用代码后才能给出,因而在针对代码的用户评论中,多为用户在使用代码后根据代码使用过程中出现的问题所进行的详细描述.如“一添加新的管理员,设置好管理员权限,登录新的管理员账号就卡死浏览器,发现是chrome浏览器会卡死,360就不会,2.0版本也有这个问题”“Swagger测试的时候,后台接受不到参数,一直运行不出来,这是什么情况”.在这些详细描述中,用户的评论或是存在多个分句,或是包含连词.通过对大量的用户评论进行分析以及结合汉语复杂句式的定义[7],本文将复杂用户评论描述为包含连词或多个分句的用户评论.并且,为了不丢失复杂用户评论中的代码质量属性信息,在对复杂用户评论进行拆分处理时,保留分句与连词之间的关系.

在前期研究中,本文采用分句的关键词完全无向图表示分句.虽然无向图可以表示出分句中的词语在位置上的聚集关系,但是无向图难以说明除位置关系以外的词语之间的语义关系.依存句法分析可以通过语句单位内词语间的依存关系揭示词语间的语义修饰关系,进而实现对语义的理解,能够较为有效地弥补单纯依靠词性等手段难以触及深层语义关系的不足[22],在情感分析[23,24]、实体抽取[25,26]、数据的结构化[27,28]等自然语言处理任务中有着广泛的应用.因此,本文在处理复杂用户评论分句时采用基于依存句法关系构建的分句有向图替代前期的无向图.

为了降低复杂用户评论的处理难度,并对复杂用户评论的代码质量属性进行有效判断,首先需要对复杂用户评论进行分句、构建分句依存句法关系有向图等预处理.在对复杂用户评论进行预处理后,形成了多个复杂用户评论分句的依存句法关系有向图.然后,应用基于依存句法关系的主题判断规则,从复杂用户评论中抽取主题,并基于初始代码质量属性特征词库识别出各主题对应的代码质量属性.接着,根据因果连词对代码质量属性的表现与表现结果进行获取.最后,基于主题处理规则,首先对同一主题的代码质量属性表现与表现结果进行处理,然后再对同一主题处理规则处理后得到的代码质量属性相关信息,应用不同主题处理规则进行处理,产生复杂用户评论的代码质量属性判断结果.

4.1 复杂用户评论的预处理

由于针对代码的用户评论多为复杂句,复杂句中糅合了多种代码质量属性相关的信息,导致直接判断复杂用户评论的代码质量属性困难,因此,为了降低文本的复杂度和处理的难度,首先对复杂用户评论进行预处理,以简化文本,帮助判断复杂用户评论的代码质量属性.

针对复杂用户评论的预处理首先需要按照标点符号(例如,逗号、分号、句号、感叹号、问号等)对复杂用户评论进行分割,得到复杂用户评论的分句.此外,由于针对代码的复杂用户评论中通常包含一些代码编译运行时编译器提示的信息,这些信息多为英文且包含一些阿拉伯数字,在对用户评论的处理中暂时不考虑英文,而用户评论中数字的处理对于分析用户评论的代码质量属性作用不大.因此,本文通过Unicode编码对文本进行过滤,使得最后得到的分句中不包含英文字符串和数字.

为了准确判断复杂用户评论中的代码质量属性,本文借助哈尔滨工业大学信息检索研究室语言技术平台[29](LTP)对分句后的复杂用户评论进行依存句法分析,获取复杂用户评论分句中存在的依存句法关系.此外,由于获取的复杂用户评论的依存句法关系是以文本形式表示,难以清晰地表示出词语之间的依存关系.因此,本文根据获取到的依存句法关系,以词语作为节点,词语之间的依存句法关系作为边,将复杂用户评论的依存句法关系抽象为依存句法关系有向图,可以清晰地表示出复杂用户评论分句中词语之间的依存句法关系.

LTP依存句法关系主要有主谓关系(SBV)、动宾关系(VOB)、间宾关系(IOB)、前置宾语关系(FOB)、兼语(DBL)、定中关系(ATT)、状中结构(ADV)、动补结构(CMP)、并列关系(COO)、介宾关系(POB)、左附加(LAD)、右附加(RAD)以及核心关系(HED)等.此外,由于核心关系(HED)指向复杂用户评论分句的核心,本文用“Root”表示分句核心关系(HED)指向的节点.以复杂用户评论“编码格式不对一直报错,而且没有注释所以看不懂.一堆堆

错误!”为例,其分句“编码格式不对一直报错”“而且没有注释所以看不懂”“一堆堆错误”的依存句法关系有向图分别如图 1 中的(1)、(2)、(3)所示.

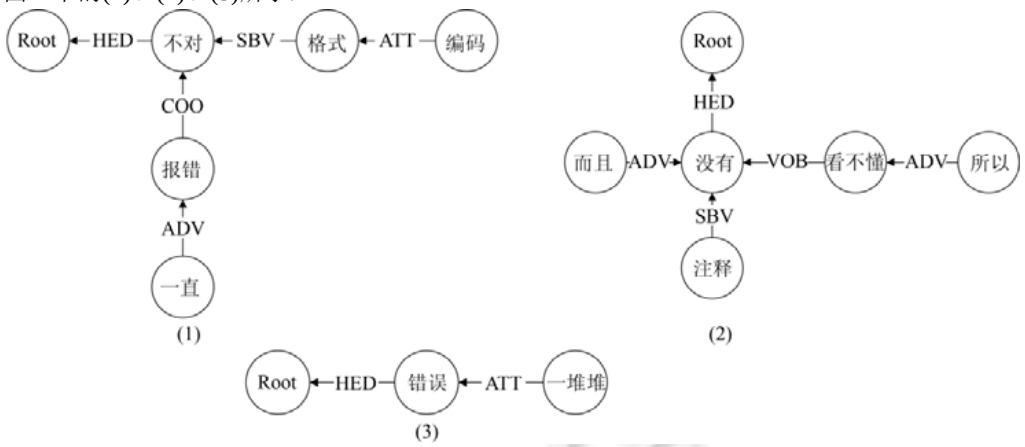

Fig.1 Example of dependency syntax relation directed graph example

图 1 依存句法关系有向图示例

由图 1 可以看出,将复杂用户评论的文本表示转换成图结构,能够清晰地表示出词语之间的关系,将有利于复杂用户评论代码质量属性的判断.

4.2 识别用户评论中与代码质量属性相关的信息

经过复杂用户评论预处理之后,复杂用户评论转换成了以分句为单位的依存句法关系有向图.通过分析大量的用户评论后发现,当分句中存在多个代码质量属性时,分句通常表现为多个主题.当在复杂用户评论分句中存在的代码质量属性不明确时,先通过抽取复杂用户评论分句的主题,再通过抽取的各主题对应的主题相关词来识别复杂用户评论中存在的代码质量属性,可以较为全面和准确地判断出复杂用户评论的代码质量属性以及与各个代码质量属性相关的词语.

4.2.1 抽取复杂用户评论的主题

为了能够较为全面地判断出复杂用户评论中的代码质量属性,本文先对预处理后的复杂用户评论的每个分句抽取主题.从长短文本的划分出发,复杂用户评论中的分句仍归属于短文本范畴,但是又比一般范畴的短文本更加短小.传统的主题抽取方法有主题模型[30,31]、改进主题模型[32,33],大多适用于一般范畴的短文本的处理,在用户评论这类短文本主题抽取上表现不佳.此外,传统的方法在抽取主题时默认主题是相互独立的,且抽取的主题相关信息粒度较粗,难以对主题相关的信息进行下一步细粒度的分析,也未对主题之间的关系进行处理.因此,为了能够对复杂用户评论分句抽取主题后进行下一步的处理和分析,本文基于依存句法关系制定了主题判断规则,并应用主题判断规则抽取复杂用户评论分句的主题.

复杂用户评论分句经过依存句法分析后会存在唯一的一个核心关系(HED),而核心关系(HED)对应的核心词可能涉及一个或多个主题.当存在多个主题时,多个主题之间一般会通过并列关系(COO)相连.此外,依存句法关系中的兼语(DBL)、定中关系(ATT)、状中关系(ADV)、动补结构(CMP)、介宾关系(POB)以及左附加(LAD)和右附加(RAD)一般是对句子的核心成分或是主谓宾成分起修饰或补充说明的作用,对句子主题个数判断的影响较小.因此,结合部分复杂用户评论分句的依存句法关系有向图的分析,本文主要围绕句子中的核心关系(HED)、并列关系(COO)以及句子主谓宾可能存在的主谓关系(SBV)、动宾关系(VOB)、间宾关系(IOB)、前置宾语关系(FOB)分析如何抽取复杂用户评论分句的主题.

在依存句法关系中,并列关系(COO)表示两个词语之间以并列形式依存,如“运行速度和内存占用”“内存占用”为“运行速度”的并列.将这种词语间的依存关系扩展到复杂用户评论分句的依存句法关系有向图中,除根节点 Root 外,以并列关系(COO)为界,并列依存的两个节点会各自构成一个有向图的子图,两个子图内的节点构成

的语义相对完整,而两个子图之间的语义则相对独立,独立、完整的语义在分句中会形成主题,从而并列关系(COO)可以由两个词语的并列扩展为两个主题的并列.因此,若以并列关系(COO)为标志,则可对复杂用户评论中的主题进行判断.本文以并列关系(COO)为核心,制定了并列关系主题判断规则,如图 2 所示.

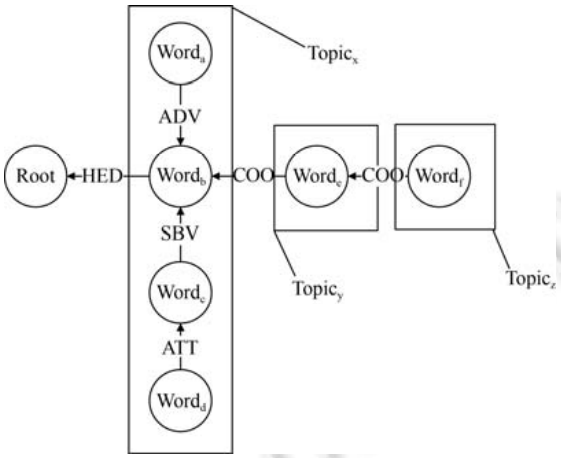

Fig.2 Topic judgment rule of coordinate relation

图 2 并列关系主题判断规则

由图 2 可以看出,两个并列关系(COO)连接的 3 个部分分别构成了主题 $Topic_x$、$Topic_y$、$Topic_z$.根据图 2 并列关系主题判断规则,可给出并列关系主题判断规则的定义如下.

并列关系主题判断规则. 复杂用户评论分句存在并列关系(COO),则以并列关系(COO)为界,并列关系(COO)连接的两部分各构成一个主题.

通过对大量的复杂用户评论分析后发现,在复杂用户评论分句中,当存在多个主题时,一般分句中的主谓成分及其修饰成分会构成一个主题的语义,而宾语成分及其修饰成分构成另一个主题的语义.而在依存句法关系中,经过依存句法分析的句子会存在唯一的一个核心关系(HED).当核心关系(HED)对应的词为动词时,该动词对应句子主谓宾中的谓语成分,与该动词构成动词与宾语的依存关系(VOB/IOB/FOB)的节点则对应句子主谓宾中的宾语成分.因此,若以核心关系(HED)以及动词与宾语的依存关系(VOB/IOB/FOB)为标志,则可对复杂用户评论中的主题进行判断.本文以核心关系(HED)为核心,制定了动宾关系主题判断规则,如图 3 所示,其中核心关系(HED)对应的核心词为 $Word_m$.

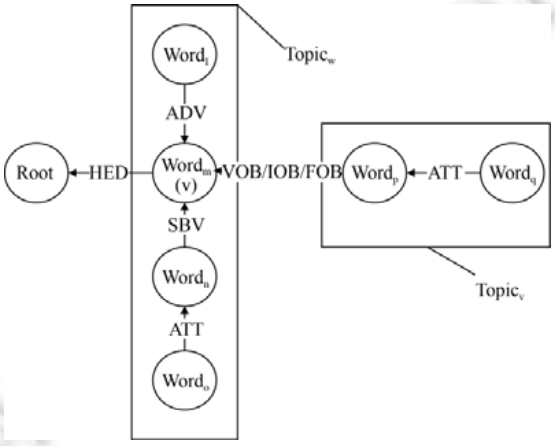

Fig.3 Topic judgment rule of verb-object relation

图 3 动宾关系主题判断规则

由图 3 可以看出,动词与宾语的依存关系(VOB/IOB/FOB)连接的两个部分分别构成了主题 $Topic_w$、$Topic_v$.根据图 3 动宾关系主题判断规则,可以给出动宾关系主题判断规则的定义如下.

动宾关系主题判断规则. 复杂用户评论核心关系(HED)对应的核心词为动词(v),核心词存在动宾关系(VOB)或间宾关系(IOB)或前置宾语关系(FOB),则以动词与宾语的依存关系(VOB/IOB/FOB)为界,动词与宾语的依存关系(VOB/IOB/FOB)连接的两部分各构成一个主题.

当复杂用户评论分句只存在一个主题时,分句中的依存句法关系有向图中的依存关系较为简单,且分句通常不满足并列关系主题判断规则和动宾关系主题判断规则.因此,对于不满足多个主题判断规则的复杂用户评论分句,则认为复杂用户评论分句只存在一个主题,从而制定了单一主题判断规则,如图 4 所示.

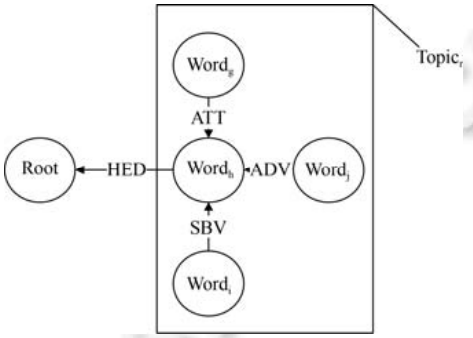

Fig.4 Topic judgment rule of single

图 4 单一主题判断规则

由图 4 可以看出,由于不存在并列关系(COO),也不满足动宾关系主题判断规则,因而只构成了一个主题 $Topic_r$.根据图 4 所示单一主题判断规则,可以给出单一主题判断规则的定义如下.

单一主题判断规则. 复杂用户评论中不存在并列关系(COO),且核心词不为动词,或核心词为动词时,核心词不存在动宾关系(VOB)或间宾关系(IOB)或前置宾语关系(FOB),则复杂用户评论只存在一个主题.

在依存句法关系有向图中,存在与各个主题直接相关的节点,这些节点构成了各个主题相关的词集(如图 1 示例中,图 1(1)的“编码”“格式”“不对”构成了一个主题相关的词集,“一直”“报错”构成了另一个主题相关的词集).在应用主题判断规则判断主题时,对于并列关系主题判断规则与动宾关系主题判断规则需要区分不同主题直接相关的节点以获取主题相关的词集.由于并列关系主题判断规则是以并列关系(COO)为界,因此,当应用并列关系主题判断规则判断复杂用户评论分句中的主题时,断开依存句法关系有向图中的并列关系(COO)有向边,此时,复杂用户评论分句的依存句法关系有向图由多个独立的子图构成,独立子图的节点则构成了主题相关的主题词集.例如图 2 中 $Topic_x$ 的主题词集为节点“Root、$Word_a$、$Word_b$、$Word_c$、$Word_d$”,$Topic_y$ 的主题词集为节点“$Word_e$”,$Topic_z$ 的主题词集为节点“$Word_f$”.同理,动宾关系主题判断规则是以动词与宾语的依存关系(VOB/IOB/FOB)为界,则当应用动宾关系主题判断规则判断复杂用户评论分句中的主题时,断开依存句法关系有向图中的动词与宾语的依存关系(VOB/IOB/FOB)有向边,复杂用户评论分句的依存句法关系有向图中的各独立子图的节点则构成了各主题相关的词集.

通过主题判断规则,可以判断复杂用户评论分句中的主题以及主题相关的词集.由于在获取复杂用户评论分句的依存句法关系时,需要相对完整的分句语义才能获取较为准确的依存句法关系.因此,在获取分句依存句法关系时,未对分句执行去停用词等操作.鉴于这种情况,在判断复杂用户评论分句中的主题与主题相关的主题词集后,需要对每个主题对应的子图进行一些剪枝操作,以去掉停用词等噪声词语,获取主题最相关的词语,即主题相关词.

通过分析复杂用户评论,复杂用户评论分句中与主题相关的依存关系主要有 HED、SBV、ADV、VOB、IOB、FOB,而与主题相关的词的词性主要为名词(n)、动词(v)、形容词(a)、副词(d).因此,综合考虑依存句法关系和词的词性制定了两类剪枝操作,如下所示.

依存句法关系剪枝. 去掉除核心关系(HED)、主谓关系(SBV)、状中关系(ADV)、动宾关系(VOB)、间宾关系(IOB)、前置宾语关系(FOB)之外的其他依存句法关系对应的有向边,以及该有向边的开始节点.

停用词剪枝. 定义“n、v、a、d”词性以外的词语节点为停用词节点,去掉停用词节点.

综上,对于满足并列关系主题判断规则与动宾关系主题判断规则的依存句法关系有向图,则对断开并列关系(COO)或动词与宾语的依存关系(VOB/IOB/FOB)后形成的有向图的子图执行剪枝操作;对于满足单一主题判断规则的依存句法关系有向图只执行剪枝操作即可.以第 4.1 节中复杂用户评论“编码格式不对一直报错,而且没有注释所以看不懂.一堆堆错误!”3 个分句依存句法关系有向图(图 1 中的(1)、(2)、(3))为例,3 个分句分别满足 3 种主题判断规则.

分句“编码格式不对一直报错”依存句法关系有向图(图 1 示例中(1))中包含并列关系(COO),满足并列关系主题判断规则,应用并列关系主题判断规则对分句进行处理,处理过程与结果如图 5 所示.节点“编码”“格式”“不对”构成一个主题,节点“报错”构成一个主题.

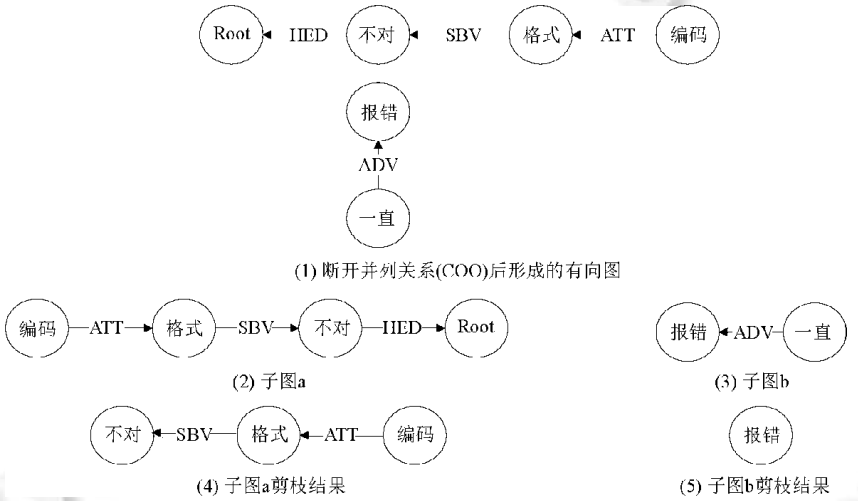

Fig.5 Example of coordinate relation topic judgment rule processing clause result

图 5 并列关系主题判断规则处理分句结果示例

分句“而且没有注释所以看不懂”依存句法关系有向图(图 1 示例中(2))中核心关系(HED)指向的词“没有”为动词,且包含动宾关系(VOB),满足动宾关系主题判断规则,因而应用动宾关系主题判断规则进行处理,处理结果如图 6 所示,节点“注释”“没有”构成一个主题,节点“看不懂”构成一个主题.

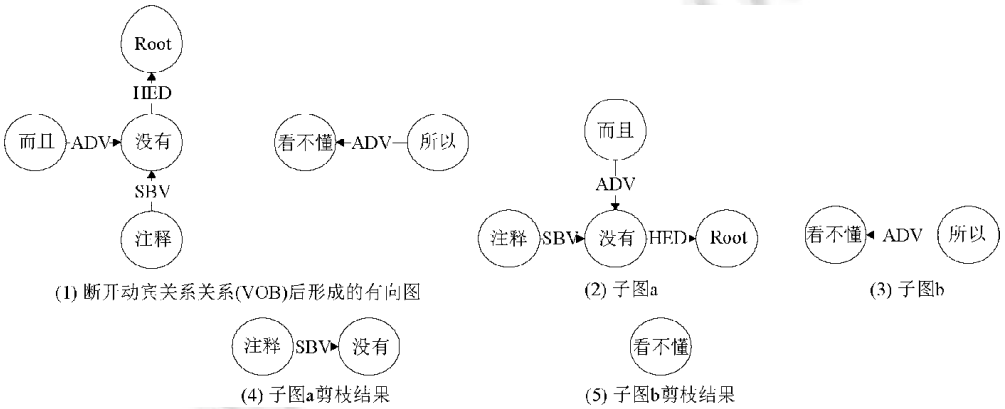

Fig.6 Example of verb-object relation topic judgment rule processing clause result

图 6 动宾关系主题判断规则处理分句结果示例

分句“一堆堆错误”依存句法关系有向图(图 1 示例中(3))中不包含并列关系(COO),也不具有动词与宾语的依存关系(VOB/IOB/FOB),符合单一主题判断规则,因而应用单一主题判断规则进行处理,只对分句依存句法关系有向图进行剪枝操作,剪枝操作结果如图 7 所示.

Fig.7 Example of single topic judgment rule processing clause result

图 7 单一主题判断规则处理分句结果示例

通过主题判断规则与剪枝操作,基于复杂用户评论分句的依存句法关系有向图抽取了复杂用户评论分句的主题以及主题相关词,将复杂用户评论的代码质量属性判断转化成了主题对应代码质量属性的识别与分析,降低了问题的复杂度,有助于判断复杂用户评论的代码质量属性.

4.2.2 识别各主题对应的代码质量属性

经过第 4.2.1 节抽取复杂用户评论的主题后,获取了复杂用户评论分句的主题和主题相关词.为了能够准确判断复杂用户评论的代码质量属性,需要识别复杂用户评论分句中各主题对应的代码质量属性.在第 3 节中,本文基于前期工作对每个代码质量属性的特征词库进行扩充,增加了代码质量属性特征词对.因此,在抽取了复杂用户评论分句的主题与主题相关词后,可以应用代码质量属性特征词对或特征词匹配各主题对应的主题相关词以识别主题对应的代码质量属性.识别复杂用户分句中各主题对应代码质量属性算法如算法 1 所示.

算法 1. 识别各主题对应的代码质量属性.

Input:TopicKeyWords=[$topicWords_1$,…,$topicWords_x$,…,$topicWords_k$]//复杂用户评论分句的主题与主题相关词,其中,$topicWords_x$=($topic_x$,$topicword_x$)为二元组,k 为分句中主题的个数

codeQualityAttributeFeatureWord={$Attribute_1$:$FeatureWordList_1$,…,$Attribute_a$:$FeatureWordList_a$,…,$Attribute_{11}$:$FeatureWordList_{11}$}(1≤a≤11))//11 个代码质量属性特征词库

Output:codeQualityAttribute//主题对应的代码质量属性列表

```
codeQualityAttribute=[]//初始化主题对应的代码质量属性列表为空
for topicWords in TopicKeyWords://遍历
    topicwords=topicWords[1]//主题相关词位于二元组的第 2 个位置
    for Attribute_a,FeatureWordList_a in codeQualityAttributeFeatureWord://遍历代码质量属性特征词库
        mixset=list(set(FeatureWordList_a).intersection(set(topicwords)))//获取主题相关词与代码质量属性特征词的交集
        T=(topic=topic_T,exp=exp_T)//初始化一个新的主题二元组,其中,topic_T,exp_T 均为空
        if len(mixset)>0://主题相关词与代码质量属性特征词的交集不为空
            topic_T=Attribute_a//识别主题对应的代码质量属性
            exp_T=topicwords//主题相关词构成主题对应代码质量属性的表现
            codeQualityAttribute.append(T)
```

通过算法 1 可以识别出复杂用户评论分句中各主题对应的代码质量属性以及代码质量属性相关的信息,此时,复杂用户评论中的多个主题可能对应同一种代码质量属性,也可能对应多种不同的代码质量属性.而在针对代码的复杂用户评论中,同种代码质量属性之间及不同代码质量属性之间均具有一定的逻辑关系.因此,通过识别复杂用户评论各主题对应的代码质量属性,有助于复杂用户评论代码质量属性的判断.

4.3 代码质量属性表现与代码质量属性表现结果识别

通过第 4.2 节的处理,基于依存句法关系有向图及主题判断规则可以识别出复杂用户评论分句中各主题对应的代码质量属性.针对代码的复杂用户评论中一般包含多个代码质量属性,且在用户评论中多个代码质量属性之间具有一定的逻辑关系.例如,第 4.1 节中的复杂用户评论分句“而且没有注释所以看不懂”,存在两个主题,两个主题对应的代码质量属性均为“可读性”.两个主题对应的代码质量属性之间由因果连词“所以”连接,则根

据因果连词语义可以判断两个主题的代码质量属性之间具有因果关系,即“没有注释”是“看不懂”的原因.因此,对于该用户评论的“可读性”代码质量属性,“没有注释”是“可读性”的表现,“看不懂”是“可读性”的表现结果.基于这种代码质量属性之间的关系,本文对用户评论中的代码质量属性表现和代码质量属性表现结果进行了定义.代码质量属性表现是指“在用户评论中,代码质量属性的具体体现.”代码质量属性表现结果是指“在用户评论中,对于同一种代码质量属性,其表现所引发的结果”.通过区分代码质量属性表现与代码质量属性表现结果,可以帮助开发者直观地获取代码质量信息.

基于代码质量属性表现与表现结果的定义,在经过第 4.2 节的处理后,各主题的主题相关词可以用来描述各主题对应的代码质量属性,则主题相关词即为代码质量属性表现.因此,经过第 4.2 节的处理,各主题对应的代码质量属性只有代码质量属性表现.根据代码质量属性表现与表现结果的定义,代码质量属性表现结果在识别之前以代码质量属性表现存在,则可以通过多个主题之间的因果关系对同一代码质量属性对应的代码质量属性表现与表现结果进行识别.本文基于主题间的因果关系识别代码质量属性表现与代码质量属性表现结果的算法如算法 2 所示.

算法 2. 基于因果关系识别代码质量属性表现与代码质量属性表现结果.

Input:Ti=(topic=$topic_{Ti}$,exp=exp_{Ti}),Tj=(topic=$topic_{Tj}$,exp=exp_{Tj})//Ti 与 Tj 之间为因果关系

Output:T=(topic=$topic_T$,exp=exp_T,res=res_T)//$topic_T$ 为主题 T 对应的代码质量属性,exp_T 为主题 T 对应的代码质量属性表现,res_T 为主题 T 对应的代码质量属性表现结果.

1 if $topic_{Ti}$==$topic_{Tj}$://Ti 与 Tj 为相同主题

2 T=(topic=$topic_T$,exp=exp_T,res=res_T)//初始化一个新的主题,其中,$topic_T$,exp_T,res_T 均为空

3 if Ti 为原因:

4 $topic_T$=$topic_{Ti}$

5 exp_T=exp_{Ti}//Ti 的代码质量属性表现 exp_{Ti} 为主题 $topic_T$ 的表现 exp_T

6 res_T=exp_{Tj}//Tj 的代码质量属性表现 exp_{Tj} 为主题 $topic_T$ 的表现 exp_T 的表现结果 res_T

7 else://Tj 为原因

8 $topic_T$=$topic_{Tj}$

9 exp_T=exp_{Tj}//Tj 的代码质量属性表现 exp_{Tj} 为主题 $topic_T$ 的表现 exp_T

10 res_T=exp_{Ti}//Ti 的代码质量属性表现 exp_{Ti} 为主题 $topic_T$ 的表现 exp_T 的表现结果

以第 4.1 节中的复杂用户评论分句“而且没有注释所以看不懂”为例,分句的依存句法关系图满足动宾关系主题判断规则.应用动宾关系主题判断规则并识别主题对应的代码质量属性后,原分句依存句法图如图 8 所示.

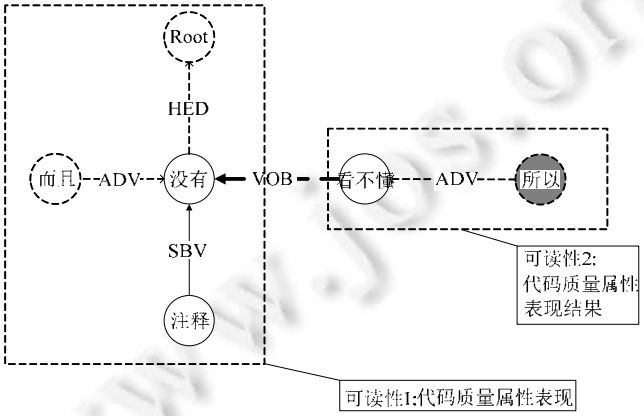

Fig.8 Example of getting the representation and the representation result of code quality attribute

图 8 获取代码质量属性表现与表现结果示例

图 8 中粗虚线箭头为主题判断规则断开的边,细虚线箭头和虚线节点为剪枝操作而去掉的边和节点,灰色

节点“所以”为因果连词中的结果连词.由图 8 可以看出,由于主题“可读性 2”对应的节点中包含因果连词中的结果连词“所以”,所以“可读性 2”对应的主题相关词“看不懂”作为分句“而且没有注释所以看不懂”对应代码质量属性“可读性”的代码质量属性表现结果,而主题“可读性 1”对应的主题相关词“没有”“注释”作为“可读性”的表现.

通过代码质量属性表现与代码质量属性表现结果的识别算法,可以识别出主题对应代码质量属性表现与表现结果.代码质量属性表现与代码质量属性表现结果可以较为清晰、完整地描述复杂用户评论中的代码质量属性,使得复杂用户评论中的代码质量属性之间初步有了较为清晰的逻辑关系,从而为分析复杂用户评论中的代码质量属性表现与表现结果,并以此判断代码质量属性奠定了基础.

4.4 分析复杂用户评论的代码质量属性表现与表现结果

通过第 4.2 节和第 4.3 节的处理,可以获得复杂用户评论中各主题对应的代码质量属性表现与表现结果.复杂用户评论或是包含多个分句,或是包含连词.由于分句之间通过标点符号进行连接,无论是否存在连词,分句之间的代码质量属性逻辑关系都为并列关系.在语言学中,连词是用来连接词与词、词组与词组或句子与句子,表示某种逻辑关系的虚词.在文本中,连词的语义不同,则连词连接的两个文本之间的逻辑关系会根据连词的并列、承接、转折、因果等不同的语义而发生改变.因而,若由连词连接的多个主题,主题间的关系会因连词的语义不同而具有不同的逻辑关系.因此,最终判断需要综合考虑分句之间存在标点符号和连词对代码质量属性逻辑关系的影响,若不考虑分句间的关系直接进行合并,就会使合并后的结果失去连贯性,甚至改变原有的含义.

综上,为了更加准确地判断复杂用户评论的代码质量属性,本文通过对复杂用户评论的研究,根据是否存在连词/标点符号、连词类别对代码质量属性逻辑关系的影响,对主题之间的逻辑关系进行了详细的定义.然后基于主题间关系提出了相同主题与不同主题的处理规则,通过主题处理规则分析复杂用户评论中的代码质量属性表现与表现结果,从而获得复杂用户评论的代码质量属性判断结果.主题间关系主要 5 种,见表 2.

Table 2 Definition of intertopic relationship

表 2 主题间关系定义

序号	主题间关系	连词	标点符号
1	并列关系(Ti,coo,Tj)	以及,和,跟,与,同,而且,还有,况且,何况 无连词	,.;!?、,.;!? 无标点符号
2	承接关系(Ti,con,Tj)	比如,此外,像,好像,如,比方,而,则,乃,就	
3	转折关系(Ti,tur,Tj)	但是,却,然而,只是,不过,不料,岂知	
4	让步关系(Ti,alt,Tj)	虽然,固然,尽管,纵然,即使	
5	因果关系(Ti,cal,Tj)	因为,由于,因此,所以,是故,是由于,以致	

由于复杂用户评论在预处理时划分为了多个分句,在判断复杂用户评论的代码质量属性时,需要综合分析复杂用户评论中各分句的主题对应的代码质量属性表现与代码质量属性表现结果.基于表 2 给出的主题间关系的定义可知,当主题间具有不同的关系时,主题对应的代码质量属性相关信息之间的关系也会发生改变.然而,针对复杂用户评论中存在多个主题时,若多个主题对应同一代码质量属性,则具有同一代码质量属性的相同主题可以根据不同主题间关系进行合并处理.如果是因果关系,可以进行代码质量属性表现与代码质量属性表现结果的划分;若为并列关系,则可以根据相同主题是否具有相同的代码质量属性表现与表现结果进行不同的处理.因此,在对复杂用户评论的代码质量属性进行判断并产生代码质量属性相关结果之前,需要对复杂用户评论中对应同一种代码质量属性的相同主题进行处理.

本文通过对复杂用户评论的研究发现,在对复杂用户评论中对应同一种代码质量属性的相同主题进行处理时,根据主题间的关系不同,主题处理主要有以下两种规则.

(1) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为并列关系、承接关系、转折关系、让步关系时,

主题处理规则有两种.

处理规则 1:若主题 Ti、Tj 对应的代码质量属性表现与表现结果无同一词,则主题 Ti、Tj 合并为同一个主题.合并后的主题对应的代码质量属性为其中一个主题对应的代码质量属性;主题对应的代码质量属性表现为 Ti、Tj 对应代码质量属性表现的并集;主题对应的代码质量属性表现结果为 Ti、Tj 对应代码质量属性表现结果的并集.

处理规则 2:若主题 Ti、Tj 对应的代码质量属性表现与表现结果具有同一词,则主题 Ti、Tj 合并为同一个主题.合并后的主题对应的代码质量属性为其中一个主题对应的代码质量属性;主题对应的代码质量属性表现为 Ti、Tj 主题对应代码质量属性表现去掉表现交集后与表现交集的并集;主题对应的代码质量属性表现结果为 Ti、Tj 主题对应代码质量属性表现结果去掉表现结果交集后与表现结果交集的并集.

(2) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为因果关系时,则主题 Ti、Tj 合并为同一个主题.将原因对应的主题的代码质量属性与代码质量属性表现作为合并后的主题的代码质量属性与代码质量属性表现,原因对应的主题的代码质量属性表现结果与另一个主题的代码质量属性表现与表现结果的并集作为合并后的主题对应的代码质量属性表现结果.

因此,基于表 2 中给出的主题间关系的定义,给出了针对同一主题的处理规则,见表 3.

Table 3 Same topic processing rule

表 3 同一主题处理规则

序号	主题间关系	主题处理规则	示例
1	并列关系 (Ti,coo,Tj)	(1) 处理规则 1 $topic_{Ti}=topic_{Tj}$,若 $exp=exp_{Ti}\cap exp_{Tj}=NULL$,$res=res_{Ti}\cap res_{Tj}=NULL$, 则 $out=\{Topic=\{topic_{Ti}\},Expression=\{exp_{Ti},exp_{Tj}\},Expression\_Result=\{res_{Ti},res_{Tj}\}\}$ (2) 处理规则 2 $topic_{Ti}=topic_{Tj}$,若 $exp=exp_{Ti}\cap exp_{Tj}\neq NULL$ 或 $res=res_{Ti}\cap res_{Tj}\neq NULL$, 则 $out=\{Topic=\{topic_{Ti}\},Expression=\{exp,exp_{Ti}\text{-}exp,exp_{Tj}\text{-}exp\},Expression\_Result=\{res,res_{Ti}\text{-}res,res_{Tj}\text{-}res\}$	(1) 编码格式不对一直报错,**而且**缩进不对 (2) 登录功能不出来,**还有**验证功能也是
2	承接关系 (Ti,con,Tj)		(1) 代码很不错,**比如**代码注释写的很清晰,很容易看懂 (2) 代码功能强大,**比如**登录功能、自动刷新功能
3	转折关系 (Ti,tur,Tj)		(1) 代码可以使用,**但是**缺少了一些包 (2) 函数方法命名可以见名知意,**但是**命名格式不太规范
4	让步关系 (Ti,alt,Tj)		(1) **尽管**没有注释,还是可以看得懂 (2) **尽管**控制流程有些小问题导致逻辑不太清楚,数据流程检查了没啥问题
5	因果关系 (Ti,cal,Tj)	(1) 处理规则 1 $topic_{Ti}=topic_{Tj}$,若 Ti 为原因, 则 $out=\{Topic=\{topic_{Ti}\},Expression=\{exp_{Ti}\},Expression\_Result=\{res_{Ti},exp_{Tj},res_{Tj}\}$ (2) 处理规则 2 $topic_{Ti}=topic_{Tj}$,若 Tj 为原因, 则 $out=\{Topic=\{topic_{Tj}\},Expression=\{exp_{Tj}\},Expression\_Result=\{res_{Tj},exp_{Ti},res_{Ti}\}$	(1) **因为**内存占用太多,代码运行速度慢 (2) 代码运行速度慢,**由于**代码内存占用太多

与同一主题判断规则不同,多个主题之间由于不具有相同的代码质量属性,不能进行合并处理.但是,不同主题间关系的不同,表现出了用户对不同主题的关注度存在差异.例如,当两个不同主题间为转折关系时,用户一般更关注转折之后的主题,则在给出复杂用户评论判断结果时,转折之后的主题对应的代码质量属性相关信息应该位于转折之前的主题对应的代码质量属性相关信息之前.因此,不能根据相同主题的处理规则对主题对应的代码质量属性相关信息直接进行处理,而应根据主题间关系的不同进行相应的处理.根据主题间关系的不同,本文对相同主题处理后得到的复杂用户评论的主题,给出主题判断的结果,对于用户更关注的主题,在判断的结果中,该主题对应的代码质量属性相关信息位置越靠前.因此,根据主题间的关系不同,不同主题处理主要有以下 4 种规则.

(1) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为并列关系、承接关系时,用户对主题 Ti、Tj 的关注度相同.在输出时,输出结果中主题 Ti、Tj 相关代码质量属性相关信息的位置按照 Ti、Tj 的先后位置进行表示.

(2) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为转折关系时,若 Tj 为转折之后的主题,则用户对 Tj 更为关注.在输出时,输出结果中主题 Tj 相关代码质量属性相关信息的位置位于主题 Ti 代码质量属性相关信息的位置之前.

(3) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为让步关系时,若 Ti 为让步的主题,则用户对非让步的主题 Tj 更为关注.在输出时,输出结果中主题 Tj 相关代码质量属性相关信息的位置位于主题 Ti 代码质量属性相关信息的位置之前.

(4) 当复杂用户评论中两个主题 Ti、Tj 的主题间关系为因果关系时,主题处理规则有两种.

处理规则 1:若主题 Ti 为原因,则用户对主题 Ti 更为关注.在输出时,输出结果中主题 Ti 相关代码质量属性相关信息的位置位于主题 Tj 代码质量属性相关信息的位置之前.

处理规则 2:若主题 Tj 为原因,则用户对主题 Tj 更为关注.在输出时,输出结果中主题 Tj 相关代码质量属性相关信息的位置位于主题 Ti 代码质量属性相关信息的位置之前.

综上,基于表 2 中给出的主题间关系的定义,给出了针对不同主题的处理规则,见表 4.

Table 4 Different topic processing rule

表 4 不同主题处理规则

序号	主题间关系	主题处理规则	示例
1	并列关系 (Ti,coo,Tj)	$topic_{Ti}\neq topic_{Tj}$, 则 out={Topic=⟨$topic_{Ti}$,$topic_{Tj}$⟩,Expression=⟨$exp_{Ti}$,$exp_{Tj}$⟩, Expression_Result=⟨$res_{Ti}$,$res_{Tj}$⟩}	项目在我的编辑器里运行不了,**而且**也没有人员维护了 代码很好用啊,复杂度没有想象的高 编码格式不对代码报错
2	承接关系 (Ti,con,Tj)		项目启动失败,**好像**是版本问题
3	转折关系 (Ti,tur,Tj)	$topic_{Ti}\neq topic_{Tj}$, 则 out={Topic=⟨$topic_{Tj}$,$topic_{Ti}$⟩,Expression=⟨$exp_{Tj}$,$exp_{Ti}$⟩, Expression_Result=⟨$res_{Tj}$,$res_{Ti}$⟩}	代码挺容易上手,**但是**有一些错误
4	让步关系 (Ti,alt,Tj)	$topic_{Ti}\neq topic_{Tj}$, 则 out={Topic=⟨$topic_{Tj}$,$topic_{Ti}$⟩,Expression=⟨$exp_{Tj}$,$exp_{Ti}$⟩, Expression_Result=⟨$res_{Tj}$,$res_{Ti}$⟩}	**尽管**没有注释,使用还是方便的
5	因果关系 (Ti,cal,Tj)	(1) 处理规则 1 $topic_{Ti}\neq topic_{Tj}$,若 Ti 为原因, 则 out={Topic=⟨$topic_{Ti}$,$topic_{Tj}$⟩,Expression=⟨exp_{Ti},exp_{Tj}⟩, Expression_Result=⟨res_{Ti},res_{Tj}⟩} (2) 处理规则 2 $topic_{Ti}\neq topic_{Tj}$,若 Tj 为原因 则 out={Topic=⟨$topic_{Tj}$,$topic_{Ti}$⟩,Expression=⟨$exp_{Tj}$,$exp_{Ti}$⟩, Expression_Result=⟨$res_{Tj}$,$res_{Ti}$⟩}	(1) **因为**没有注释,代码不好用 (2) 代码出现了错误,检查了才发现是**由于**代码内存溢出

*topic=⟨$topic_{Tj}$,$topic_{Ti}$⟩表示 $topic_{Tj}$ 的重要性大于 $topic_{Ti}$,位于元组的第 1 个位置

以第 4.1 节中的复杂用户评论“编码格式不对一直报错,而且没有注释所以看不懂!一堆堆的错误!”为示例,通过第 4.1 节~第 4.3 节处理后得到复杂用户评论主题对应的代码质量属性相关信息.然后,根据表 3 和表 4 的主题处理规则,可以给出复杂用户评论的代码质量属性判断结果,见表 5.

由表 5 可以看出,主题处理规则可以对复杂用户评论的主题进行有效的处理.

Table 5 Example of judgment result
表 5 判断结果示例

处理过程	处理结果
复杂用户评论	编码格式不对一直报错,而且没有注释所以看不懂!一堆堆的错误!
预处理用户评论 (第 4.1 节)	分句 1:编码格式不对一直报错 分句 2:而且没有注释所以看不懂 分句 3:一堆堆错误
抽取分句主题 (第 4.2.1 节)	分句 1:[主题 1:编码 格式 不对;主题 2:报错;] 分句 2:[主题 1:没有 注释;主题 2:看不懂;] 分句 3:[主题 1:错误;]
识别分句各主题对应的代码质量属性 (第 4.2.2 节)	分句 1:[主题 1:topic=编程规范性,exp=编码 格式 不对;主题 2:topic=可靠性,exp=报错;] 分句 2:[主题 1:topic=可读性,exp=没有 注释;主题 2:topic=可读性,exp=看不懂;] 分句 3:[主题 1:topic=可靠性,exp=报错;]
识别代码质量属性表现与表现结果(第 4.3 节)	分句 1:[主题 1:topic=编程规范性,exp=编码 格式 不对,res=NULL;主题 2:topic=可靠性,exp=报错;res=NULL;] 分句 2:[主题 1:topic=可读性,exp=没有 注释,res=看不懂;] 分句 3:[主题 1:topic=可靠性,exp=错误,res=NULL;]
同一主题处理(第 4.4 节)	[主题 1:topic=编程规范性,exp=编码 格式 不对,res=NULL;主题 2:topic=可靠性,exp=[报错][错误];res=NULL;主题 3:topic=可读性,exp=没有 注释,res=看不懂;]
不同主题处理(第 4.4 节)	[主题 1:topic=〈编程规范性,可靠性,可读性〉,exp=〈[编码 格式 不对],[报错][错误],[没有注释]〉,res=〈,,[看不懂]〉;]

4.5 动态扩充初始的代码质量特征词库

在经过第 4.4 节主题判断规则处理后,可以给出复杂用户评论中存在的多种代码质量属性的表现与表现结果等相关信息的判断结果.在得到的复杂用户评论的代码质量属性表现与表现结果中,包含了代码质量属性特征词库中的特征词,以及一些与特征词共同出现的主题相关词,这些主题相关词与其他标识代码质量的特征词共同出现更能准确地体现某种代码质量属性.因此,为了能够更加准确地标识用户评论的代码质量属性以及对新的数据进行准确处理,本文采用半监督自学习的方式,以名词、动词、形容词、副词为限制条件,对识别出的复杂用户评论的代码质量属性表现与表现结果对应的主题相关词分别抽取主题对应的关键词,将抽取的主题关键词按照其在复杂用户评论中的位置重新组合,作为该主题对应的代码质量属性的特征词对,加入到初始的代码质量属性特征词库中.例如,“代码运行失败了”与“怎么就运行困难了呢”对应的主题关键词分别为“代码运行 失败”与“运行 困难”,虽然都包含特征词“运行”,但是分别可以准确对应“可测试性”与“易用性”.因而,将新的特征词对加入特征词库后,在下一次使用扩充后的代码质量属性特征词库时,优先匹配新加入的特征词对可以更加准确地判断代码质量属性.此外,在初始代码质量属性特征词库中,包含了单个特征词与少量的特征词对.而在动态扩充后,特征词库中包含了丰富的特征词对,可以充分利用有限代码质量信息对特征词库进行扩展.扩充代码质量属性特征词库的算法如算法 3 所示.

算法 3. 动态扩充初始的代码质量属性特征词库的算法.

Input:Expression-ResultList={$expressionresult_1$,…,$expressionresult_i$,…,$expressionresult_o$}($1\leqslant i\leqslant o$)//复杂用户评论代码质量属性相关信息列表,其中,$expressionresult_i$=($Attribute_i$,$Expression_i$,$Result_i$)为三元组,o 为复杂用户评论中代码质量属性个数

codeQualityAttributeFeatureWord={$Attribute_1$:$FeatureWordList_1$,…,$Attribute_a$:$FeatureWordList_a$,…,$Attribute_{11}$:$FeatureWordList_{11}$}($1\leqslant a\leqslant 11$)//11 个代码质量属性特征词库

Output:codeQualityAttributeFeatureWord//扩充后代码质量属性特征词库

```
1 for expressionresult in Expression-ResultList://遍历复杂句的代码质量属性相关信息列表
2     Attribute=expressionresult[0]//代码质量属性
3     expression=expressionresult[1]//代码质量属性表现
4     result=expressionresult[2]//代码质量属性表现结果
5     expFeawordpair=getFeaturewordPair(expression)//抽取代码质量属性表现中的主题关键词构建表现
```

特征词对

resFeawordpair=getFeaturewordPair(result)//抽取表现结果中的主题关键词构建表现结果特征词对

for $Attribute_a$, $FeatureWordList_a$ in codeQualityAttributeFeatureWord://遍历代码质量属性特征词库

if expFeawordpair not in $FeatureWordList_a$://判断表现特征词对是否在某个代码质量属性特征词表中

codeQualityAttributeFeatureWord[$Attribute_a$].append(expFeawordpair)//不存在,则加入到对应的代码质量属性列表中

if resFeawordpair not in $FeatureWordList_a$://判断表现结果特征词对是否在某个代码质量属性特征词表中

codeQualityAttributeFeatureWord[$Attribute_a$].append(resFeawordpair)//不存在,则加入到对应的代码质量属性列表中

例如,以第 4.4 节中复杂用户评论“编码格式不对一直报错,而且没有注释所以看不懂!一堆堆的错误!”处理后的结果为例.复杂用户评论中的代码质量属性及其表现与表现结果有“〈编程规范性,[编码 格式 不对],[]〉,〈可靠性,[报错][错误],[]〉,〈可读性, [没有 注释], [看不懂]〉”,则将“编码 格式 不对”加入到“编程规范性”的代码质量属性特征词库中,将“没有 注释”加入到“可读性”的代码质量属性特征词库中,“看不懂”与“报错”“错误”分别在“可读性”与“可靠性”对应的代码质量属性特征词库中已经存在,则不用再加入.

5 实验结果与分析

本文的实验数据来源于专业开发者社区 CSDN(https://www.csdn.net)和开源中国推出的代码托管平台 Gitee(https://gitee.com)爬取的针对代码的用户评论,抽取了其中包含多个标点符号的复杂用户评论 4 366 条及 654 条不带标点符号的复杂用户评论.然后,应用本文方法对 5 020 条复杂用户评论进行了相关实验.

5.1 基于主题判断规则抽取复杂用户评论的主题

为了能够判断出复杂用户评论中存在的代码质量属性,本文基于依存句法关系提出了主题判断规则.先通过抽取复杂用户评论中可能存在的主题,再通过主题对应的代码质量属性的识别判断复杂用户评论中可能存在的代码质量属性.本文在应用主题判断规则抽取复杂用户评论的主题后,对主题个数以及主题对应的代码质量属性的判断结果,与人工标注进行了对比.通过分析对比结果发现,主题判断结果主要有 4 种类型.表 6 中给出了 4 种主题判断的结果类型及其对应的用户评论数量和占比,以及相应的示例.

经过分析我们发现,对于表 6 中的类型 3,由于主题判断规则不全,导致应用当前的主题判断规则对部分复杂用户评论进行判断时,主题划分失败,从而影响主题的判断.如类型 3 示例中虽然包含承接连词“就”,但在经过依存句法分析后,示例对应的依存句法关系有向图不能满足并列关系和动宾关系的主题判断规则,只能使用单一主题判断规则进行处理,因而,只判断出了一个主题,从而漏判了主题.对于类型 4,由于主题判断规则的不完善,在对部分复杂用户评论的主题进行判断时,分句中的主题在语义理解上应为一个主题,但是由于符合当前的主题判断规则,从而被判断为两个主题.如类型 4 用户评论示例分句“登录出现很多问题”,语义理解上应是登录功能出现了问题,然而在基于主题判断规则进行主题抽取时,由于满足动宾关系主题判断规则而被划分为两个主题.综上,虽然主题判断规则不是十分完善,但是可以对复杂用户评论中存在的代码质量属性进行较为全面的判断.

此外,为了说明主题判断规则抽取复杂用户评论主题的有效性,本文使用传统的隐含狄利克雷分布主题模型(LDA)抽取复杂用户评论主题.对应表 6 中 4 种主题判断结果类型,本文方法与 LDA 主题模型主题判断结果的对比情况如图 9 所示.

Table 6 Topic judgement result
表 6 主题判断结果

序号	主题判断结果	用户评论数量	占复杂用户评论总数比例	示例
1	主题个数正确,主题对应的代码质量属性正确	2 322	0.46	用户评论:这样使用乱码,代码没有测试成功 人工标注主题:2 个(编程规范性 1:使用乱码;可测试性 1:代码没有测试成功) 方法判断主题:2 个(编程规范性 1:使用乱码;可测试性 1:代码没有测试成功)
2	主题个数正确,部分主题错误	957	0.19	用户评论:不支持 div 整除操作!会报错,这个可以修改吗? 人工标注主题:2 个(功能性 1:不支持整除操作;可靠性 1:报错;) 方法判断主题:2 个(可移植性 1:不支持整除操作;可靠性 1:报错;)
3	主题个数错误,部分主题漏判	724	0.15	用户评论:这个用更新的版本代码就报错了 人工标注主题:2 个(可维护性 1:更新版本代码;可靠性 2:报错;) 方法判断主题:1 个(可靠性 1:报错;)
4	主题个数错误,部分主题多判	1 017	0.2	用户评论:运行工程,登录后出现很多问题,不维护该项目了? 人工标注主题:2 个(可维护性 1:不维护;功能性 1:登录出现问题;) 方法判断主题:3 个(可维护性 1:不维护;功能性 1:登录;可靠性:出现问题;)

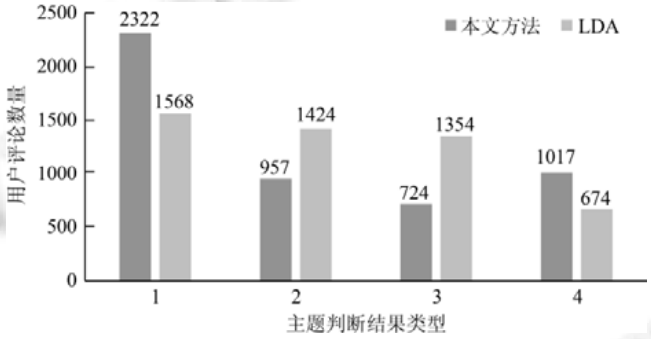

Fig.9 Comparison of topic judgment results between the proposed method and LDA
图 9 本文方法与 LDA 的主题判断结果对比

由图 9 可以发现,对应表 6 中类型 1 与类型 2,在主题个数判断正确时,本文方法在主题对应的代码质量属性判断正确的用户评论数量上高于 LDA,而在主题对应的代码质量属性部分主题判断错误的用户评论数量上低于 LDA.此外,当主题个数判断错误时,对应表 6 中的类型 3,本文方法漏判主题的用户数量比 LDA 要少,而对应于表 6 中的类型 4,本文方法多判主题的用户评论数量多于 LDA.通过对出现的结果进行分析发现,出现这种结果的主要原因是由于短文本的特征稀疏,导致 LDA 在进行短文处理时产生误判和漏判主题的情况较多,多判主题的情况较少.而本文的主题判断规则基于依存句法关系,不同的依存句法关系对应的词都可作为主题判断的特征,主题特征相对较多,不同主题特征的覆盖率较高.因而,主题全部判断正确的用户评论数量较多,同时也会导致多判的情况比漏判的情况更多.综上,虽然本文方法在主题多判结果中高于 LDA,但在主题判断正确结果上的表现优于 LDA.因此,虽然本文方法未完全判断出复杂用户评论中所有存在的代码质量属性,但是可以对复杂用户评论中存在的代码质量属性进行较为完整的判断.

为了进一步分析本文提出的主题判断规则的主题判断准确性,本文应用主题判断的准确率进行评估.对应用本文方法获得的主题个数以及主题对应的代码质量属性的判断结果,与人工标注的结果进行对比,根据对比

结果是否一致计算出主题判断规则的准确率.对应表 6 中的类型 1 与类型 2,在方法判断与人工标注的主题个数相同的前提下,对比人工标注,进行主题判断准确率的计算,具体计算如式(1)、式(2)所示.

$$OneAccTopic = \frac{x}{y} \tag{1}$$

$$AccTopic = \frac{1}{n}\sum_{i=1}^{n} OneAccTopic_i \tag{2}$$

其中,x 为某条复杂用户评论本文方法判断和人工标注的相同主题个数,y 为该条复杂用户评论人工标注的主题个数,$OneAccTopic_i$ 为第 i 条复杂用户评论的主题判断的准确率,n 为复杂用户评论总数.

通过计算,主题判断的准确率为 0.66,表明本文方法的主题判断规则可以对复杂用户评论中的主题进行较为有效的判断.

5.2 复杂用户评论的代码质量属性判断

为了进一步说明本文方法在复杂用户评论代码质量属性及其相关信息判断的有效性,本文将方法最终判断的代码质量属性相关结果与人工标注进行对比,分别计算了复杂用户评论在代码质量属性、代码质量属性表现和代码质量属性表现结果判断的准确率.准确率的计算采用取交集的方式,首先分别对每条用户评论的每种代码质量属性结果的准确率进行计算,然后再对所有用户评论的代码质量属性准确率的计算结果取平均值,将平均准确率作为应用本文方法处理得到该代码质量属性结果的准确率.具体计算如式(3)、式(4)所示.

$$OneAccAtt = \frac{Iset}{Pset} \tag{3}$$

$$AccAtt = \frac{1}{n}\sum_{j=1}^{n} OneAccAtt_j \tag{4}$$

其中,$Iset$ 为某条复杂用户评论对于某种代码质量属性结果人工标注和本文方法得到结果的交集中元素的个数,$Pset$ 为对于某条复杂用户评论的某种代码质量属性结果中应用本文方法得到的元素个数,$OneAccAtt_j$ 为第 j 条复杂用户评论的某种代码质量属性结果的准确率,n 为复杂用户评论总数.

对比人工标注,应用本文方法处理得到的代码质量属性相关结果的准确率见表 7.

Table 7 Accuracy of three results of code quality attributes

表 7 代码质量属性相关结果的准确率

判断对象	准确率
代码质量属性	0.86
代码质量属性表现	0.83
代码质量属性表现结果	0.74

由表 7 可知,通过应用本文提出的复杂用户评论的代码质量属性判断方法,代码质量属性、代码质量属性表现、代码质量属性表现结果判断的准确率可以达到 0.86、0.83 和 0.74,与代码质量属性及代码质量属性表现判断准确率相比,代码质量属性表现结果判断的准确率较低.通过对判断的代码质量属性表现结果进行分析发现,其原因是由于代码质量属性表现与表现结果的识别只是基于连词而产生的因果关系,导致对不具有因果连词产生的因果关系的代码质量属性表现与表现结果难以识别,因而代码质量属性表现结果判断的准确性较弱.综上,虽然代码质量属性表现结果的准确率相对较低,但是可以表明本文针对复杂用户评论的代码质量属性的判断方法可以对复杂用户评论的代码质量属性及代码质量属性相关信息进行较为准确的判断.

此外,对比初始代码质量属性特征词库与经过处理后扩充的特征词库,扩充后的代码质量属性特征词库增加了一定数量的特征词对.本文在前期工作中代码质量属性特征词库为 63 个特征词,本文在实验开始时有 152 个特征词对,实验结束后扩充后的代码质量属性特征词库中有 653 个特征词对,新增了 501 个特征词对.为了验证扩充的特征词对的有效性,应用本文第 4 节提出的针对复杂用户评论的代码质量属性判断方法,在第 4.2 节识别用户评论与代码质量属性相关信息的过程中,使用扩充后包含 653 个特征词对的代码质量属性特征词库对

同一批复杂用户评论的代码质量属性进行了判断,得到的代码质量属性、代码质量属性表现、代码质量属性表现结果判断的准确率分别为 0.88、0.89 和 0.79,比应用初始代码质量属性特征词库的准确性(见表 7)有所提升.而且,由于扩充的特征词主要为特征词对,在代码质量属性的描述上,特征词对比单个词能够对代码质量属性进行更为具体的描述,因而在代码质量属性表现与代码质量属性表现结果的准确率上提升较大.综上,实验结果表明,扩充代码质量属性特征词库可以扩大代码质量属性的信息量,能够更加准确地判断出复杂用户评论的代码质量属性信息.

综上所述,通过实验结果可以看出,应用本文提出的针对复杂用户评论的代码质量属性判断方法可以对复杂用户评论的代码质量属性进行有效判断.本文所提出的特征词对、主题判断规则与主题处理规则均是可扩展的.我们在实验中发现,在部分用户评论中使用现有的规则无法判断出代码质量属性.例如,用户评论“这个用最新版本的库就报错了”(主题 1 可移植性与主题 2 可靠性)与“怎么登录不了浏览器也卡死了呢”(主题 1 功能性与主题 2 运行性能)均包含了两个主题,然而,在当前第 4.2.1 节中定义主题判断规则中,由于其依存句法关系有向图不满足并列关系与动宾关系主题判断规则,只能通过单一主题判断规则进行判断,则会漏判其中一个主题,导致不能判断出用户评论中全部的质量属性.产生这种现象的原因是,在这类用户评论的依存句法关系有向图中,每条用户评论的两个主题之间的主题词集均是由于依存关系而相互交织在一起,导致一些规则难以进行清晰的抽象及定义.因此,为了进一步提高代码质量属性判断的准确率,需要通过持续地对更多的用户评论进行分析、抽象和总结,不断扩展现有规则.此外,代码质量的度量应包括代码静态分析与动态测试等,而本文提出的方法只属于代码质量度量的一部分,可以与代码的静态分析、动态测试等方法相结合,对代码质量进行综合判断,从而为用户和开发人员提供客观的代码质量判断结果.

6 结束语

本文提出的针对复杂用户评论的代码质量属性判断方法,解决了复杂用户评论中同时存在多个代码质量属性时代码质量属性判断的问题.针对当前相关研究中对复杂用户评论中存在代码质量属性只能进行较为片面的判断,本文从 11 个静态和动态代码质量属性出发,基于依存句法关系提出了主题判断规则,首先,通过对复杂用户中存在的主题进行判断,找出复杂用户评论中可能存在的主题.然后,对判断出的主题进行其对应的代码质量属性的识别.通过主题判断和主题对应的 11 个代码质量属性的识别,能够从多角度对复杂用户评论中的代码质量属性进行较为全面的判断.此外,针对当前相关研究方法中存在代码质量属性判断不准确的问题,本文提出了主题处理规则,对判断出的复杂用户评论中可能存在的主题进行同一主题合并和不同主题的处理.通过对相同主题与不同主题相应的处理,能够较为准确地给出复杂用户评论的代码质量属性相关信息的判断结果.最后,为了能够充分利用有限的代码质量信息,根据代码质量属性相关信息的判断结果,对代码质量属性特征词库进行了扩充.

在本文的实验中,由于只是基于因果关系获取主题对应的代码质量属性的表现与表现结果,导致部分复杂用户评论中的代码质量属性的表现结果并未识别,无法进入下一步的处理,代码属性表现结果判断准确率较低.此外,由于主题判断规则的不完善,存在对部分复杂用户评论的主题判断过多或主题漏判的情况,继而影响代码质量属性的判断.因此,为了提高复杂用户评论代码质量属性判断的准确性,下一步将对获取代码质量属性表现与代码质量属性表现结果的方法和基于依存句法关系的复杂用户评论主题判断规则进行完善.

References:

[1] Berkhan D. Software component score: Measuring software component quality using static code analysis. In: Proc. of the Computational Science and Its Applications—ICCSA 2015. 2015,9159:63–72. [doi: 10.1007/978-3-319-21413-9_5]

[2] Mamun MAA, Berger C, Jörgen H. Correlations of software code metrics: An empirical study. In: Proc. of the IWSM Mensura 2017: The 27th Int’l Workshop on Software Measurement and the 12th Int’l Conf. on Software Process and Product Measurement, 2017.

[3] Kaur A, Nayyar R. A comparative study of static code analysis tools for vulnerability detection in C/C++ and JAVA source code. Procedia Computer Science, 2020,171:2023–2029. [doi: 10.1016/j.procs.2020.04.217]

[4] Ruiz-Rube I, Person T, Dodero JM, Mota J, Mota JM, Sánchez-Jara JM. Applying static code analysis for domain-specific languages. Software & Systems Modeling, 2020,19(1):95–11. [doi: 10.1007/s10270-019-00729-w]

[5] Pagano D, Brügge B. User involvement in software evolution practice: A case study. In: Proc. of the 2013 Int'l Conf. on Software Engineering. San Francisco: IEEE Press, 2013. 953–962.

[6] Xu HY, Jiang Y. Code quality recognition and analysis based on user's comments. Computer Science, 2020,47(6):44–50 (in Chinese with English abstract).

[7] Li AP, Qiu P, Duan LG. Document sentiment orientation analysis based on sentence weighted algorithm. Journal of Chinese Computer Systems, 2015,36(10):2252–2256 (in Chinese with English abstract).

[8] 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]

[9] Duan WJ, Jiang Y. Defect recognition of APP software based on user feedback. Computer Science, 2020,47(6):44–50 (in Chinese with English abstract).

[10] Wang DX, Wang Q. Trustworthiness evidence supporting evaluation of software process trustworthiness. Ruan Jian Xue Bao/Journal of Software, 2018,29(11):3412–3434 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5291.htm [doi: 10.13328/j.cnki.jos.005291]

[11] Venkatasubramanyamrd RD, Sowmya GR. Why is dynamic analysis not used as extensively as static analysis: An industrial study? In: Proc. of the 1st Int'l Workshop on Software Engineering Research and Industrial Practices—SER&IPs 2014. 2014. 24–33.

[12] Liu ZG, Chen XR. A entity-action relationship model for text clustering. Journal of Chinese Information Processing, 2018,32(5):22–30 (in Chinese with English abstract).

[13] Liu ZG, Chen XR. Research on argument relationship model based in syntactic analyses. Journal of Nanjing University (Natural Science), 2019,55(6):1010–1019 (in Chinese with English abstract).

[14] Mao TT, Lv XQ, Zhou Q, Liu Y. Manual annotation approach to Chinese complex sentences by using bottom-up and top-down. Journal of Chinese Computer Systems, 2016,37(4):716–721 (in Chinese with English abstract).

[15] Ye ZL, Jia Z, Yin HF, *et al*. Research on multi-domain natural language question understanding. Computer Science, 2017(6):216–221 (in Chinese with English abstract).

[16] Swain D, Tambe M, Ballal P, Dolase V, Agrawal K, Rajmane Y. Lexical text simplification using WordNet. Communications in Computer and Information Science, 2019,1046:114–122. [doi: 10.1007/978-981-13-9942-8_11]

[17] Siddharthan A. Syntactic simplification and text cohesion. Research on Language & Computation, 2006,4(1):77–109. [doi: 10.1007/s11168-006-9011-1]

[18] Andreasen E, Gong L, Møller A, Pradel M, Selakovic M, Sen K, Staicu C. A survey of dynamic analysis and test generation for JavaScript. ACM Computing Surveys, 2017,50(5):66:1–36. [doi: 10.1145/3106739]

[19] Selakovic M, Pradel M, Karim R, Tip F. Test generation for higher-order functions in dynamic languages. Proc. of the ACM on Programming Languages, 2018,2:27:1–27. [doi: 10.1145/3276531]

[20] Huang PJ, Yang MQ. Research and application of static metrics for code quality. Computer Engineering and Applications, 2011, 47(23):61–63 (in Chinese with English abstract).

[21] Zheng RJ. Computer Software Testing Technology. Beijing: Tsinghua University Press, 1992. 31–35 (in Chinese).

[22] Yu Y, Chen L, Jiang JD, Zhao NX. Research on the selection of Chinese patent candidate term based on dependency syntax parsing. Library and Information Service, 2019,63(18):109–118 (in Chinese with English abstract).

[23] Agarwal B, Poria S, Mittal N, Gelbukh A, Hussain A. Concept-level sentiment analysis with dependency-based semantic parsing: A novel approach. Cognitive Computation, 2015,7(4):487–499. [doi: 10.1007/s12559-014-9316-6]

[24] Feng C, Liao C, Liu ZR, Huang HY. Sentiment key sentence identification based on lexical semantics and syntactic dependency. Acta Electronica Sinica, 2016,44(10):2471–2476 (in Chinese with English abstract).

[25] Gan LX, Wan CX, Liu DX, Zhong Q, Jiang TJ. Chinese named entity relation extraction based on syntactic and semantic features. Journal of Computer Research and Development, 2016,53(2):284–302 (in Chinese with English abstract).

[26] Wan CX, Gan LX, Jiang TJ, Liu DX, Liu XP, Liu Y. Chinese named entity implicit relation extraction based on company verbs. Chinese Journal of Computers, 2019,42(12):2795–2820 (in Chinese with English abstract).

[27] Tian CY, Chen DH, Wang M, Le JJ. Structured processing for pathological reports based on dependency parsing. Journal of Computer Research and Development, 2016,52(12):2669–2680 (in Chinese with English abstract).

[28] Luo SL, Han L, Pan LM, Wei C. Construction method of Chinese sentential semantic structure. Journal of Beijing Institute of Technology, 2015(1):110–117.

[29] Che WX, Li ZH, Liu T. LTP: A Chinese language technology platform. In: Proc. of the 23th Int'l Conf. on Computational Linguistics—COLING 2010. Beijing, 2010. 23–27.

[30] Chen Q, Zhang L, Jiang J, Huang XY. Review analysis method based on support vector machine and latent Dirichlet allocation. Ruan Jian Xue Bao/Journal of Software, 2019,30(5):1547–1560 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5731.htm [doi: 10.13328/j.cnki.jos.005731]

[31] Zhang YT, Wan CX, Liu XP, Jiang TJ, Liu DX, Liao GQ. Mining unstructured economic indicators based on PSP_HDP topic model. Ruan Jian Xue Bao/Journal of Software, 2020,31(3):845–865 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5898.htm [doi: 10.13328/j.cnki.jos.005898]

[32] Gao HY, Liu JW, Yang SX. Identifying topics of online healthcare reviews based on improved LDA. Trans. of Beijing Institute of Technology, 2019,39(4):427–434 (in Chinese with English abstract).

[33] Zhang ZY, Huo WG. A topic text network construction method based on PL-LDA model. Complex Systems and Complexity Science, 2017,14(1):52–57, 110 (in Chinese with English abstract).

附中文参考文献:

[6] 徐海燕,姜瑛.基于用户评论的代码质量识别与分析.计算机科学,2020,47(3):41–47.

[7] 李爱萍,邸鹏,段利国.基于句子情感加权算法的篇章情感分析.小型微型计算机系统,2015,36(10):2252–2256.

[8] 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]

[9] 段文静,姜瑛.基于用户反馈的 APP 软件缺陷识别.计算机科学,2020,47(6):44–50.

[10] 王德鑫,王青.支持软件过程可信评估的可信证据.软件学报,2018,29(11):3412–3434. http://www.jos.org.cn/1000-9825/5291.htm [doi: 10.13328/j.cnki.jos.005291]

[12] 刘作国,陈笑蓉.面向文本聚类的实体-动作关联模型研究.中文信息学报,2018,32(5):22–30.

[13] 刘作国,陈笑蓉.汉语句法分析中的论元关系模型研究.南京大学学报(自然科学版),2019,55(6):1010–1019.

[14] 毛婷婷,吕学强,周强,刘殷.融合从底向上与自顶向下的中文复杂句人工标注方法.小型微型计算机系统,2016,37(4):716–721.

[15] 冶忠林,贾真,尹红风.多领域自然语言问句理解研究.计算机科学,2017,44(6):216–221.

[20] 黄沛杰,杨铭铨.代码质量静态度量的研究与应用.计算机工程与应用,2011,47(23):61–63.

[21] 郑人杰.计算机软件测试技术.北京:清华大学出版社,1992.31–35.

[22] 俞琰,陈磊,姜金德,赵乃瑄.基于依存句法分析的中文专利候选术语选取研究.图书情报工作,2019,63(18):109–118.

[24] 冯冲,廖纯,刘至润,黄河燕.基于词汇语义和句法依存的情感关键句识别.电子学报,2016,44(10):2471–2476.

[25] 甘丽新,万常选,刘德喜,钟青,江腾蛟.基于句法语义特征的中文实体关系抽取.计算机研究与发展,2016,53(2):284–302.

[26] 万常选,甘丽新,江腾蛟,刘德喜,刘喜平,刘玉.基于协陪义动词的中文隐式实体关系抽取.计算机学报,2019,42(12):2795–2820.

[27] 田驰远,陈德华,王梅,乐嘉锦.基于依存句法分析的病理报告结构化处理方法.计算机研究与发展,2016,52(12):2669–2680.

[30] 陈琪,张莉,蒋竞,黄新越.一种基于支持向量机和主题模型的评论分析方法.软件学报,2019,30(5):1547–1560. http://www.jos.org.cn/1000-9825/5731.htm [doi: 10.13328/j.cnki.jos.005731]

[31] 张奕韬,万常选,刘喜平,江腾蛟,刘德喜,廖国琼.基于 PSP_HDP 主题模型的非结构化经济指标挖掘.软件学报,2020,31(3):845–865. http://www.jos.org.cn/1000-9825/5898.htm [doi: 10.13328/j.cnki.jos.005898]

[32] 高慧颖,刘嘉唯,杨淑昕.基于改进 LDA 的在线医疗评论主题挖掘.北京理工大学学报,2019,39(4):427–434.

[33] 张志远,霍纬纲.一种基于 PL-LDA 模型的主题文本网络构建方法.复杂系统与复杂性科学,2017,14(1):52–57,110.

徐海燕(1996－),女,学士,CCF 学生会员,主要研究领域为软件工程,软件质量保证与测试.

姜瑛(1974－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件质量保证与测试,云计算,大数据分析,智能软件工程.