软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2021,32(4):1023−1038 [doi: 10.13328/j.cnki.jos.006228]

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

融合代码与文档的软件功能特征挖掘方法*

沈 琦[1,2], 钱 莹[1,2], 邹艳珍[1,2], 伍仕骏[1,2], 谢 冰[1,2]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](北京大学 信息科学技术学院,北京 100871)

通讯作者: 邹艳珍, E-mail: zouyz@pku.edu.cn

摘 要: 在软件复用过程中,简洁、清楚的软件功能自然语言描述是帮助复用者快速了解待复用软件项目/代码库的前提和基础.但当前开源软件往往缺乏高质量的软件功能说明文档,使得这一过程变得更加复杂和困难.为此,提出了一种融合代码与文档的软件功能特征挖掘方法.该方法以动宾短语的形式描述软件功能特征,通过迭代挖掘软件源代码和以 Stack Overflow 讨论帖为代表的软件文档,自动提取开源软件的功能特征描述,并构造了层次化的软件功能特征视图.在针对多个开源软件项目的实验中,该方法可覆盖官方文档中列举的 95.38%的软件功能.挖掘结果中语句和功能特征的准确率分别达到了 93.78%和 92.57%.对比现有工作 TaskNav 和 APITasks,该方法在平均准确率上分别提升了 28.78%和 11.56%.

关键词: 软件复用;软件功能特征;开源软件;自然语言编程接口

中图法分类号: TP311

中文引用格式: 沈琦,钱莹,邹艳珍,伍仕俊,谢冰.融合代码与文档的软件功能特征挖掘方法.软件学报,2021,32(4):1023−1038. http://www.jos.org.cn/1000-9825/6228.htm

英文引用格式: Shen Q, Qian Y, Zou YZ, Wu SJ, Xie B. Fusing code and documents to mine software functional features. Ruan Jian Xue Bao/Journal of Software, 2021,32(4):1023−1038 (in Chinese). http://www.jos.org.cn/1000-9825/6228.htm

Fusing Code and Documents to Mine Software Functional Features

SHEN Qi[1,2], QIAN Ying[1,2], ZOU Yan-Zhen[1,2], WU Shi-Jun[1,2], XIE Bing[1,2]

[1](Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

[2](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: In the process of software reuse, users need concise and clear natural language description of software functions to understand the candidate software project quickly. However, current open source software often lacks high-quality documentation, which makes this process even more complex and difficult. This study proposes a novel functional feature mining approach combining code and documentation. It describes functional features in the form of verb phrases, automatically extracts functional features by iterately mining source code and software documents such as Stack Overflow, associates corresponding API usage example for each functional feature, and builds hierarchical functional feature view for uses finally. The experiments are set on several open source software and its related heterogeneous data, the results show that the functional features generated by the proposed approach cover 95.38% of the functions in official documentation, and the proposed approach achieves 93.78% and 92.57% accuracy for mining sentences and functional features respectively. Compared to two existing tools TaskNav and APITasks, the proposed approach improves the accuracy by 28.78% and 11.56% separately.

* 基金项目: 国家自然科学基金(61972006); 国家杰出青年科学基金(61525201)

Foundation item: National Natural Science Foundation of China (61972006); National Natural Science Fund for Distinguished Young Scholars (61525201)

沈琦和钱莹为本文共同第一作者.

本文由“面向领域的软件系统构造与质量保障”专题特约编辑潘敏学教授、魏峻研究员、崔展齐教授推荐.

收稿时间: 2020-09-13; 修改时间: 2020-10-26; 采用时间: 2020-12-19; jos 在线出版时间: 2021-01-22

Key words: software reuse; software functional feature; open source software; natural language interface

当前软件开发过程中通常需要大量复用已有的代码库或开源软件[1−3].在复用过程中,用户的开发任务需求习惯使用自然语言描述,并且通常一个开发任务的代码实现涉及到多个 APIs 调用.因此,了解待复用软件项目/代码库可提供的功能特征十分重要[4−6].一方面,这些功能特征可以辅助用户快速将开发任务映射到需要调用的 APIs;另一方面,功能特征介绍可以帮助用户进一步明确自己的任务需求,防止需求描述不准确造成的时间浪费.

一般来说,了解软件的功能特征可以通过阅读该软件提供的功能描述文档.然而,不同于传统商业软件,开源软件通常缺少高质量的功能描述文档,在使用过程中常常存在下述问题[7−9]:(1) 文档介绍的功能较少,且描述不准确.由于人工编写软件功能描述文档需要大量的时间,很多时候,开发者在文档中只对一些典型的、基础的软件功能进行了介绍,且文档中的功能描述方式可能与复用者的功能需求存在一定差别.譬如在一些软件功能文档中,索引描述的可能是高层次的软件功能类别而非具体的功能特性;某些功能文档中仅仅是按照 API 的名称进行排序、组织,这使得在官方文档中自动定位到复用者需要的信息并不是一件简单的事情;(2) 文档与软件版本不一致.持续更新软件文档是一件很困难的事情.尤其是当前软件的迭代周期很短,软件功能不断变化,让软件文档始终与最新的软件功能特性、API 使用场景等保持一致是一件需要耗费大量时间和精力的事情.因此,当开发人员想要使用一些新的软件功能特性或者将 API 应用到一些新的场景中时,很可能无法在官方文档中找到对应的说明信息.

为此,需要研究提出一种软件功能特征的自动挖掘方法.该方法能够利用开源软件的各类相关资源,自动地挖掘出软件项目/代码库的功能特征,并对其进行有效组织、整理以方便复用者进行检索和浏览.在现有工作中,研究者们已经通过分析处理各类软件资源(包括官方文档、邮件列表、缺陷报告、问答网站等等),提出了多种软件功能特征/描述挖掘方法[10−12].但是,这些方法的主体采用自然语言处理技术,很大程度上依赖软件文档中大量重复出现的软件功能描述语句,或借助人工制定的单词白名单[13]、文档标题等结构信息[14]来提取软件功能描述.由于不同软件项目间的领域词汇和文档结构通常存在差异,因此方法难以在跨项目场景下保证高准确率.此外,一些代码注释或代码摘要生成[15,16]的工作采用模板填充或机器学习的思路,结合代码中 API 的上下文自动生成代码的自然语言注释或摘要.此类工作虽然同样能以自然语言功能描述的形式帮助开发者理解示例代码,进而学习软件项目,但相较于抽取式的功能描述挖掘方法,其生成描述文本的可读性较差.而对于一个新发布的软件项目来说,收集整理大量用于学习的使用示例也是一件非常耗时、耗力的事.

针对上述问题,本文提出了一种融合代码与文档的软件功能特征挖掘方法.该方法以动宾短语形式描述一项软件功能特征,通过迭代挖掘软件的源代码和 Stack Overflow 开发交流记录,实现更为准确的软件功能特征提取,并构建了多维的、层次化的软件功能特征视图.对比现有工作,本文的主要贡献包括:

(1) 提出了一种软件概念的迭代抽取方法,将源代码中的类名作为种子概念,通过迭代挖掘得到新的软件概念集,有效地指导了软件文档中软件功能特征动宾短语的挖掘;

(2) 提出了一种融合源代码与 Stack Overflow 交流记录的软件功能特征挖掘方法,以软件概念为核心,提高了软件功能特征挖掘的准确率、覆盖率和效率;

(3) 在大量软件项目数据上对本文方法进行了实验并开源了实验结果.实验结果表明,本文方法获取的软件功能特征可以覆盖 Apache POI 项目官方文档中列举的 95.65%的软件常用功能,从软件项目 Stack Overflow 交流记录中进行软件功能特征挖掘的准确率达到了 93.78%.

本文第 1 节介绍我们的方法框架以及基本概念.第 2 节具体介绍所提出的功能特征挖掘方法.第 3 节通过具体开源软件数据对本文方法进行实验评估.第 4 节介绍相关工作并进行讨论.第 5 节总结全文.

1 方法框架

软件功能是软件系统或部件定义的目标或特征动作[10].传统上,软件文档中会用一段话来描述软件功能(包

括功能的目标或动作,前提和后验条件、执行序列和输入输出等)[11,13].本文为了帮助用户进行快速浏览和定位,采用动宾短语的形式描述软件的功能特征(functional feature)[17].软件功能特征的操作、对象、约束条件等组成分别对应于动宾短语中的动词、作为宾语的名词短语和介词短语等.譬如开源软件项目 Apache POI 具有“set up print area”“convert a huge.csv file to excel”等功能特征.这种动宾短语描述的功能特征形式简洁、明确,易于开发人员理解,直接对应到复用该软件可以实现的编程目标或特征动作.

软件功能特征本质上是一段软件功能描述文本的代称.但软件功能的概念源自需求领域,因此功能特征的粒度一直是研究者们关注的重点.在本文工作中,将软件功能特征分为两个层次.

(1) 基本功能特征:使用一个软件 API 即可完成的功能特征.对复用者来说,软件功能的基本单位是 API,则一个 API 对应的软件功能称为基本软件功能特征;

(2) 复合功能特征:需要组合调用多个 APIs 才能完成的功能特征.通常来说,一个开发任务对应的软件功能需要多个 APIs 联合实现.譬如,实现“iterate over cells”功能需要使用循环调用 Sheet 对象的 getRow 和 Row 对象的 getCell 方法协作完成.

基于这种区分,本文在软件功能特征挖掘过程中对软件的源代码和 Stack Overflow 问答文档进行了综合分析,首先从源代码中提取基本功能特征和软件核心概念,并以此为基础指导从软件文档中精确提取复合软件功能特征.具体的方法流程如图 1 所示.

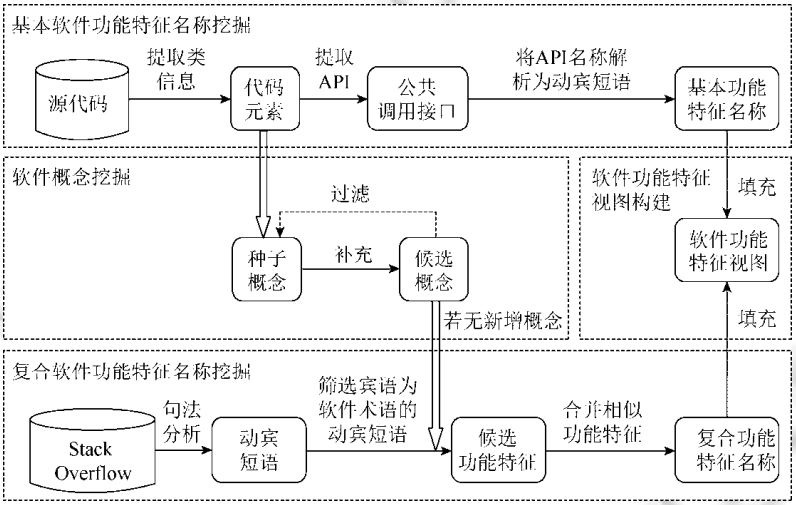

Fig.1 The framework of our approach

图 1 本文的方法框架

具体的方法流程主要包括:

基本软件功能特征挖掘.以软件项目源代码作为数据源,解析、提取其中的类、方法、字段等信息,形成代码元素集合.然后,依据访问控制修饰符筛选代码元素集合中的 API,形成公共调用接口集合,并将每个 API 名称解析、修正、还原为动宾短语,从而得到软件的基本软件功能特征集.

软件概念挖掘.软件代码中包含软件功能特征最核心的基础概念.为此,本文采用迭代扩充的思路,最初以代码元素(主要是类名)作为种子概念,然后设计启发式规则提取新的概念补充到概念集合中,形成候选概念集合;对当前的候选概念集合进一步加以过滤,保障概念扩充的准确性.重复上述过程,直到某次扩充时没有新增概念为止,即得到软件项目的核心概念集合.

复合软件功能特征挖掘.以 Stack Overflow 的软件问答文档作为数据源,首先对文档中的自然语言文本片段进行句法分析,得到动宾短语集合.然后,利用上文得到的软件核心概念集合,筛选出与相关的动宾短语作为候选复合软件功能特征名称.最后,合并相似的软件功能特征名称,从而得到复合软件功能特征名称集.

软件功能特征视图构建.挖掘基本功能特征和复合功能特征之间的关联关系.将功能特征名称中宾语对象之间的关联关系与源代码中类之间的继承关系相对应,将来源于同一个讨论帖的功能特征之间建立关联,构建多维的、层次化的功能特征视图.

为了进一步阐明上述框架,我们通过一个具体实例来展示本文功能特征抽取的基本过程.图 2 展示了一个 Stack Overflow 讨论帖中对 POI 软件一次使用的情况.这个帖子中的动宾短语“apply background color for the rows in excel sheet(由红色下划线标注)”是 POI 软件项目的一项功能特征,其作用是为 excel 文件中的行设置背景颜色.但同时帖子中包含大量的其他动宾短语,那些蓝色下划线标注的动宾短语与功能特征无关,绿色下划线标注的动宾短语则与红色下划线标注的短语含义相似.因此,如何准确地挖掘出红色下划线标注的功能特征短语是本文面临的主要技术挑战.基于上述框架,本文首先从 POI 的源代码中提取代码元素集合,获得“color”这个基本概念.然后,使用软件概念集合与基于软件开发和问答论坛场景建立的停用词表筛选动宾短语.由于动词“apply”不属于停用词表,因此,“apply background color for the rows in excel sheet”成为候选功能特征.同理,通过过滤掉蓝色下划线标注的动宾短语,留下红色下划线和绿色下划线标注的动宾短语作为候选功能特征.最后,合并相似的候选功能特征,由于红色下划线和绿色下划线标注的动宾短语相似度很高,为避免冗余,合并为一个软件功能特征.因此,从这个讨论帖中提取出的软件功能特征为“apply background color for the rows in excel sheet”.

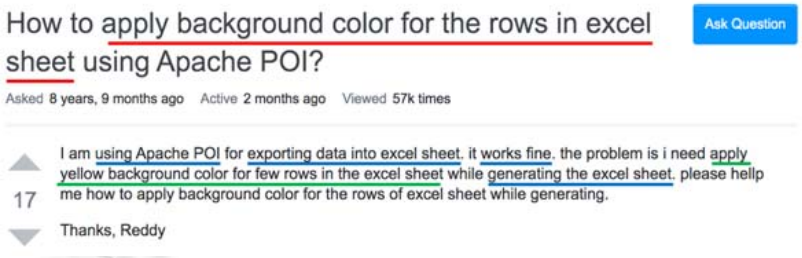

Fig.2 A software functional feature on Stack Overflow Q&A post

图 2 Stack Overflow 软件问答文档中的功能特征举例

2 软件功能特征挖掘方法

基于上述框架,这里对本文提出的融合代码与文档的软件功能特征挖掘方法进行详细介绍.

2.1 基本软件功能特征挖掘

基本软件功能特征是指由单个 API 即可实现的软件功能.本文利用软件项目源代码中的信息来挖掘基本软件功能特征名称,挖掘过程主要分为代码元素的提取、公共调用接口的提取和 API 名称的解析这 3 个步骤.

1) 代码元素的提取

本文中代码元素特指软件源代码中的类名、方法名和字段名.为了得到软件的代码元素信息,本文首先使用 javalang 来解析软件项目源代码,获得代码元素集合.对于一个软件项目的源代码,javalang 深度优先遍历软件目录,将源代码文件解析成抽象语法树(abstract syntax tree,简称 AST),然后提取 AST 中的类和方法元素,形成代码元素集合.针对每个 Java 类,获得类名称、类中有字段和方法,类可能与其他类有关系(比如继承自某个类或者实现了某个接口);针对每个 Java 方法,获得方法名称、参数和返回值以及说明该方法访问权限的修饰符等.这些抽取出的代码元素信息接下来不仅仅是基本软件功能特征名称的来源,还会作为种子概念成为软件概念挖掘的数据源.

2) 公共调用接口的提取

虽然基本软件功能特征名称与 API 的粒度一致,但是并非所有 API 都是基本软件功能特征.有些 API 仅能够在该软件项目的同一包或同一类中被调用,因此不能为软件项目的复用者所用.本文提取出软件项目的公共

调用接口,作为对应于基本软件功能特征的 API.

在 Java 软件项目中,可以使用访问控制修饰符来说明访问权限,以保护对类、变量、方法和构造方法的访问.软件功能是软件用户有权限调用的 API,这些 API 至少需要是对软件项目中所有类可见的.因此,本文提取出以“public”关键字修饰的方法作为公共调用接口,也就是基本软件功能特征名称所对应的 API.

3) API 名称的解析

本文中软件功能特征名称是以动宾短语的形式存在的.我们根据源代码中的 API 命名特点,将 API 名称解析为动宾短语,从而得到基本软件功能特征名称.

在软件项目源代码中,API 命名有以下两个主要特点:(1) 在表现形式上,软件项目中的 API 名称基本上都遵循驼峰式命名法.(2) 在 API 名称的自然语言含义方面,软件项目中的方法命名一般是动词或动词短语,与方法所施加的对象共同组成动宾短语,即动词+名词的形式.一般可以通过方法名称直接获知该方法实现什么样的功能.为此,本文将 API 名称按照驼峰式命名法进行切分,即可得到候选动宾短语.在此基础上,进一步修正、还原动宾短语,得到基本功能特征名称.其中,修正的过程是指:将动宾短语中的单词首字母还原为小写字母.如上文所述,驼峰式命名法利用单词的首字母大写来表示单词的起始,因此,需要将单词中的字母全部还原为小写字母.例如,开源软件项目 Apache POI 的“setBorderColor”方法切分后成为“set Border Color”短语,还需要将短语中的词语还原为原形,变成“set border color”.还原的过程是指:将动宾短语中的单词还原为原形.动宾短语中有些单词使用的是复数形式,例如,开源软件项目 Apache POI 的“collectValues”方法,将其切分并还原为小写字母后得到“collect values”动宾短语,这里面的“values”为复数形式,需要还原为原形“value”.本文利用自然语言处理工具包 spaCy 将单词还原为原形.

经过上述步骤,即可得到格式统一的基本软件功能特征名称.以开源软件项目 Apache POI 中的方法名称为例,“getDocument”“setBorderColor”“getFirstColumn”等在表现形式方面都是按照小驼峰式命名的,在自然语言含义方面分别表达了获取文件(get document)、设置边界颜色(set border color)、获取第 1 列(get the first column)的含义,使开发人员可以一目了然这些方法的功能.

2.2 软件核心概念挖掘

本文从源代码的代码元素中进一步提取软件概念,用以辅助复合软件功能特征名称的挖掘.这里,软件概念指的是与软件项目相关的名词,是软件功能潜在的操作对象.软件概念的挖掘采用迭代扩充的算法,流程中主要分为软件概念的扩充和软件概念的过滤这两个步骤.初始时,由代码元素中的类名组成种子概念集合.

2.2.1 软件概念的扩充

本文基于软件项目源代码中的类间关系和类中的内容扩充种子概念,从而得到软件项目的候选概念集合.以种子概念为基准,针对以 Java 为例的面向对象编程语言,本文设计了如表 1 所示的软件概念的扩充规则.

Table 1 Rules for extracting new software terms

表 1 新软件概念的抽取规则

规则	结构			新增软件概念
	源	关系	目标	
R1	C_1	Extends	C_2	C_2
R2	C_1	Implements	I	I
R3	C_1	Has get/Set method	M_2	m_2

注:C=class,M=method,I=interface,m is M without get/set prefix

本文设计的软件概念的扩充规则具体说明如下.

R1 基于 extends 关键字进行扩充.该关键字用来表示两个类之间的继承关系.如果类 C_1 继承自类 C_2,则 C_2 就是一个新的软件概念.例如,“Picture extends Shape”语句表示 Picture 类继承自 Shape 类,以种子概念“Picture”为基准,就得到了新的软件概念“Shape”.

R2 基于 implements 关键字进行扩充.该关键字意味着一个类实现了一个接口,也用来表示继承关系.如果类 C_1 实现了接口 I,那么 I 就是一个新的软件概念.例如,“BooleanFunction implements Function”语句表示

BooleanFunction 类实现了 Function 接口,以种子概念“BooleanFunction”为基准,就得到了新的软件概念“Function”.

R3 基于类中的 get/set 方法进行扩充.如果类 C_1 中有 get/set 方法,那么 get/set 方法名中去掉“get”“set”前缀的对象就是一个新的软件概念.例如,HSSFCell 类中包含 getCellType 和 setCellType 方法,那么去掉“get”“set”前缀就得到了新的软件概念“CellType”.

根据以上 3 条规则,将种子概念集合进行扩充,就得到了候选软件概念集合.如果当前的候选软件概念集合与种子概念集合相比没有新增数据,即可结束软件概念的挖掘流程,当前的候选软件概念集合即为该软件项目概念挖掘的最终结果;否则,按照后文所述对候选软件概念集合进行过滤,并将过滤结果重新作为种子概念集合,迭代挖掘出软件概念.

2.2.2 软件概念的过滤

上文得到的候选软件概念集合中可能会存在与具体的软件项目无关或者无意义的词语,因此,每次依据扩充规则得到新的候选软件概念集合之后,就需要对该候选集合进行过滤.过滤主要依据以下两条规则.

1) 去除表示基本数据类型的候选概念.以 Java 编程语言为例,它具有 char、boolean、byte、short、int 等基本数据类型.这些基本数据类型与具体的软件项目并无关联,是编程语言中的普遍概念,因此,在挖掘软件项目相关概念时应当被过滤掉.

2) 去除无意义的词语.例如,在开源软件项目 Apache POI 的 CTOfficeArtExtension 类中包含 getAny 和 setAny 方法,依据上一节中的扩充规则,会得到新的软件概念“Any”.但是,“Any”意为任何一个,并不是一个有明确意义的词汇,因此应当被过滤掉.

根据以上两条规则对候选软件概念集合进行过滤,过滤结果即可作为下一轮迭代挖掘的种子概念集合.重复上述扩充-过滤的过程,直到概念集合不再变换为止.

2.3 复合软件功能特征挖掘

基于软件概念,本文选取目前最大的问答论坛 Stack Overflow 作为具体数据来源,从软件文档问答文档中提取相关动宾短语.具体的挖掘过程主要分为文本提取与句法分析、动宾短语筛选和软件功能特征名称合并这 3 个步骤.

2.3.1 文本提取与句法分析

在 Stack Overflow 讨论帖中,有问题、回答和讨论 3 部分内容.问题中描述了作者的疑问,也是讨论帖的主题.回答可能会有多个,作者可以选择接受其中的一个高质量的正确的回答,未被接受的回答通常都是不正确的或者描述不清楚的.讨论中的语言较为日常化,而且涉及的内容并不是针对问题的解答.因此,为了获得正确且高质量的数据,本文只考虑问题和作者接受的回答中的文本片段.

为了提取出文本片段,本文利用 BeautifulSoup 来提取 Stack Overflow 讨论帖的文本内容.BeautifulSoup 是一个可以将 HTML 文件转换成复杂的树形结构并进而提取数据的 Python 库.接下来,利用自然语言处理工具包 spaCy 来提取文本内容中的动宾短语.具体包括 3 个步骤:(1) 将文本片段解析为 AST.(2) 遍历 AST,提取出动宾短语.(3) 将动宾短语中的动词还原成原形.动词可能为第 3 人称单数或者过去式或者现在分词,需要还原为统一的形式.由此,我们得到了每个讨论帖中的动宾短语,从而得到整个软件项目的动宾短语集合.

2.3.2 动宾短语的筛选

在上述动宾短语集合中,并非所有动宾短语描述的都是软件功能.通常在如下两种情况下动宾短语与软件功能无关.

(1) 动宾短语描述的可能是问答论坛场景中的动作,比如在 Stack Overflow 讨论帖中,“cause problem”“believe me”“share answer”“follow tips”等动宾短语描述的都是由人来完成的事情,都是与软件项目的功能无关的.

(2) 动宾短语使用一些没有明确指向意义的词语或者并非软件项目特定场景下的词语,使得短语的意义模糊不清或者与特定软件项目无关.比如,如果动宾短语中使用“this”“it”“them”等代词作为宾语成分,那么无法确

定这些代词的具体含义;如果动宾短语中使用“answer”“post”等问答论坛中的常用词语或者“module”“version”等软件开发过程中的普遍用语作为宾语成分,那么也是与特定的软件项目无关的.

由以上分析可知,在软件功能无关的动宾短语中,宾语成分都是与特定软件项目无关的词语.因此,需要对动宾短语集合进行筛选,仅保留与软件项目有关的动宾短语,作为候选复合软件功能特征名称.考虑到代码标示符中常使用单词的缩写等形式,直接将宾语成分与软件概念进行匹配会误筛一些有意义的动宾短语.因此,除了词根化后的单词完全匹配外,本文参照已有工作[16]考虑了如下 3 种情况.

① 软件概念是宾语成分的首字母缩写,如“nn”可以匹配“neural network”.

② 软件概念是宾语成分的前缀,如“doc”可以匹配“document”.考虑到过短的前缀会造成大量的错误匹配,实现中规定前缀的长度至少为 3.

③ 去除软件概念和宾语成分的公共前缀/后缀单词后满足情况①或情况②,如“neural net”可以匹配“neural network”.

值得注意的是,有些动宾短语的宾语成分并不属于特定软件项目的概念集合,也不是软件开发过程中的普遍用语,而是软件领域的通用概念.例如,“border”“style”“method”等,以这些通用概念为宾语成分的动宾短语也应当被认为与该软件项目有关.本文基于软件开发和问答论坛场景建立停用词表,并建立软件领域的通用概念词表,利用这两个词表和上文挖掘出的软件概念集合来筛选动宾短语,提取出宾语为特定软件项目相关概念的动宾短语.由此,就得到了候选复合软件功能特征名称集合.以开源软件项目 Apache POI 为例,“add cell”“fill pattern”“resize picture”等动宾短语中的宾语“cell”“pattern”“picture”都是该软件项目的相关概念,可以作为候选复合软件功能特征.

2.3.3 软件功能特征合并

候选复合软件功能特征名称集合中的元素都是与软件项目相关的动宾短语,都描述了软件功能.但是,其中的动宾短语可能意义相同或者相近,有两种情况:(1) 两个动宾短语的谓语、宾语和约束成分都相同,含义也相同,只是“a”“an”“the”等修饰成分有差异.例如,“adjust the column width”和“adjust column width”都表达了“设置列的宽度”的含义,“add hyperlink”和“add a hyperlink”都表达了“添加超链接”的含义.(2) 两个动宾短语使用的动词不同,但含义相近.例如,“use color”和“apply color”都表达了“使用颜色”的含义,“hold value”和“keep value”都表达了“保持值不变”的含义.考虑到上述情况,为了避免同义或者近义复合软件功能特征名称同时存在产生冗余,本文对候选复合软件功能特征名称进行了合并.

候选复合软件功能特征名称都是以软件概念为宾语的动宾短语.软件概念集合中的各个名词都有其特定的含义,不涉及同义或者近义的问题.因此,在对动宾短语进行合并时,仅需要考虑对于宾语相同的短语,其中的动词成分是否相似、宾语的约束成分是否相同.

本文利用自然语言处理工具集 NLTK(Natural Language Toolkit)的 WordNet 来计算动词相似度.在 WordNet 中,名词、动词、形容词和副词各自被组织成一个同义词的网络,可以通过计算同一词性的不同词语之间的距离来得知它们的语义相似度.该距离为[0,1]之间的一个数值,当该数值为 1 时,反映了两个词语在一个同义词集合中,即两个词语同义.由此,合并候选复合软件功能特征名称的步骤如下.

(1) 将动宾短语按照宾语成分分为不同的小集合,每个小集合中的动宾短语都具有相同的宾语成分.

(2) 对一个小集合中的动宾短语,按照以下两条规则判断任意两个动宾短语是否相似.如果两个动宾短语相似,则只保留一个动宾短语,并将与这两个动宾短语相关的信息合并.具体判断规则是:首先,去掉动宾短语中的“a”“an”“the”等修饰成分,只保留谓语、宾语、约束成分,字母全部转化成小写形式.如果此时两个动宾短语完全一致,那么就认为两个动宾短语相似;其次,计算两个动宾短语的谓语的相似度,即利用 WordNet 计算两个动词是否属于一个同义词集合.如果两个动宾短语经过规则(1)的处理只有谓语不同,其他成分都相同,而且两个动宾短语的谓语是同义词,那么就认为两个动宾短语相似.

按照上述步骤,即可完成相似动宾短语的合并,于是就可以得到软件项目的复合软件功能特征集合.

2.4 软件功能特征视图构建

软件项目具有丰富的功能特征.以开源软件项目为例,本文方法为Apache POI挖掘了上万个软件功能特征.如果将软件功能特征简单地存储到集合中,无疑会对开发人员的浏览和检索带来困难.因此,本文对软件功能特征进行层次化的组织和展示,形成具有对象层和功能特征层两层结构的软件功能特征视图,并挖掘视图中元素之间的关联关系,以方便开发人员检索相关的功能特征.

软件功能特征视图包含两个层次:对象层将软件功能特征施加的对象组织在一起,功能特征层将与某个对象相关的软件功能特征组织在一起.其中,基本软件功能特征所对应的对象是功能特征所属的类的名称,复合软件功能特征所对应的对象是功能特征名称中的宾语成分.以开源软件项目Apache POI为例,“get anchor”“apply transform”等基本功能特征是“DrawShape”类中的公共调用接口,因此,这些基本软件功能特征对应的对象层元素就是类名“DrawShape”.该软件项目拥有“access cell”“add cell”“arrange cell”等复合软件功能特征名称,这些功能特征的宾语成分都是“cell”,因此都应当与“Cell”对象对应.

软件功能特征视图中包含3种关联关系:(1) 具有功能(“has_function”)是对象与功能特征之间的关联关系,表示对象具有相关的功能特征,由上文所述的功能特征与对象的对应规则来建立;(2) 继承(“inherits”)是对象层中的关联关系,表示一个对象继承自另一个对象,由类间的继承关系来建立;(3) 来源相同(“from_sam_post”关联)是功能特征层中的关联关系,表示两个功能特征来源相同,由功能特征来源于的Stack Overflow讨论帖是否相同来建立.

图3展示出:(1) 软件功能特征视图的结构示例;(2) 用户搜索浏览软件功能的工具界面.可以看到,软件功能特征视图以一个软件概念为根节点,以树形结构展开软件概念上的功能(如从“Sheet”概念展开到“create sheet”)以及继承的子概念(如从“Sheet”概念展开到“HSSFSheet”).用户在使用网页工具浏览软件功能特征时,首先指定软件项目,接着可以用关键词检索对应的软件概念和功能特征,工具提供简单的补全机制并推荐3个相关概念和功能,用户可以点击检索项进入软件概念页面或功能特征页面.在概念页面中,除了给出概念名称和所对应的类/接口名,还展示了当前概念所对应的功能特征(对应“has_function”关联)以及继承当前类/实现当前接口的概念(对应“inherits”关联).在功能特征页面,给出了功能特征名称以及Stack Overflow上下文中的代码片段,同时给出了在同一个讨论贴中出现的其他功能特征(对应“from_same_post”关联).

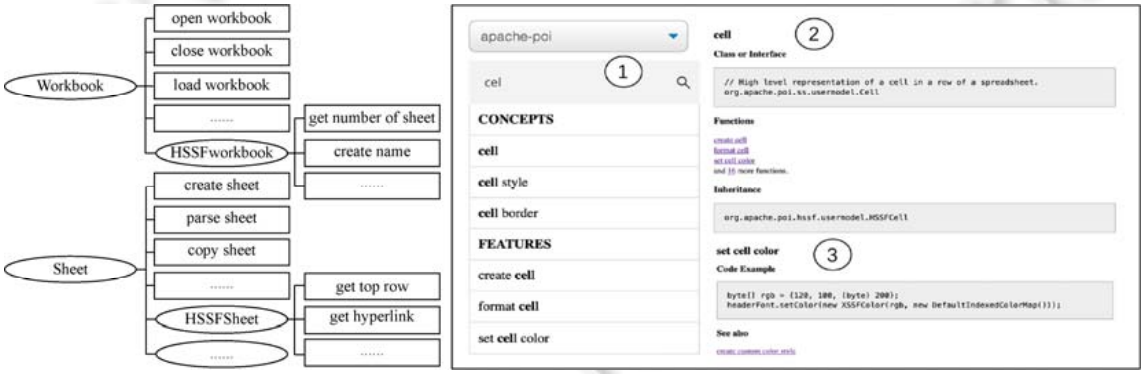

Fig.3 Example and UI of software functional feature view

图3 软件功能特征视图示例和工具界面

3 实验分析

为了验证本文工作,我们选取若干开源软件代码库,收集整理了其软件源代码、官方文档以及大量的Stack Overflow数据进行实验.在实验中,重点研究和分析以下几个问题.

本文方法获取的软件功能特征具有什么特点?是否覆盖了一个软件项目的基本功能?为此,我们针对特定

软件项目或代码库进行了功能特征挖掘,并将挖掘结果与该软件已有的官方功能文档进行了对比实验.

针对 Stack Overflow 文档中出现的功能特征描述,本文方法进行软件功能特征提取的准确率如何?为此,我们采用人工标注方式对挖掘结果进行了分析,给出了客观评价.

与现有工作相比,本文方法的优势在哪里?从源代码中提取的软件概念对软件功能特征挖掘过程有哪些改进?为此,我们在实验中对比了不同策略的挖掘结果,并进行了实际数据上的验证.

3.1 实验数据

本文选取 3 个开源软件项目数据进行了实验验证,包括 Apache POI、Eclipse JDT 和 JFreeChart.选择这 3 个软件项目的主要原因是:(1) 这些软件项目开源,容易获取到软件的源代码,并且这些软件项目分别覆盖不同类型的应用领域:Microsoft 文件处理、Java 代码解析和图表绘制,可以了解不同领域代码的特点;(2) 这些软件项目使用比较广泛,可以从 Stack Overflow 问答论坛获取到大量的讨论数据;(3) 这些软件项目的官方文档质量较好,可以获取到更多对比实验数据.例如,Apache POI 软件项目的功能文档(https://poi.apache.org/components/spreadsheet/quick-guide.html)以列表的形式展示了常见的功能特征及其代码示例;Eclipse JDT 在 Java 学习网站 ProgramGreek 上面提供了 Eclipse JDT Tutorials(https://www.programcreek.com/2011/01/best-java-development-tooling-jdt-and-astparser-tutorials/),列出了常见的功能特征及其代码示例;(4) 实验评估人员较为熟悉这些软件项目,也便于人工标注标准数据集.

实验数据的具体情况见表 2.每个软件项目的实验数据包括源代码、功能文档和讨论帖这 3 部分.其中,软件源代码来源于 GitHub 上软件项目的最新版本;功能文档来源于官方网站或者大型学习网站;讨论帖来源于 Stack Overflow 问答论坛.我们分别以“apache-poi”“eclipse-jdt”“jfreechart”为标签下载讨论帖,并对讨论帖按照投票数(vote)进行排序,选用投票数大于 0 的讨论帖作为实验数据.表 2 展示了源代码中包含的类和 API 的数量、功能文档中包含的功能特征和 API 的数量以及 Stack Overflow 讨论帖数量.其中,软件项目 JFreeChart 没有功能文档,故未作统计.

Table 2 Statistics for our dataset

表 2 实验数据集情况

软件项目	源代码		功能文档		Stack Overflow 讨论帖	
	类	API	功能特征	API	收集下载	实验分析
Apache POI	1 320	11 943	46	635	8 149	3 483
Eclipse JDT	1 744	23 479	19	341	805	489
JFreeChart	609	7 741	/	/	2 584	1 471

3.2 RQ1:对比官方文档中功能特征的覆盖情况

基于上述方法和实验数据,本文对 Apache POI、Eclipse JDT 和 JFreeChart 这 3 个软件项目的功能特征进行了挖掘.挖掘结果见表 3.以 Apache POI 为例,在挖掘过程中我们共计得到了 11 176 个软件功能特征,其中包括 8 763 个基本软件功能特征和 2 413 个复合软件功能特征.挖掘过程中,我们从源代码中获取的代码元素包括 1 320 个类和 11 943 个方法,迭代挖掘得到 2 072 个软件概念,从 Stack Oveflow 讨论帖中提取了 12 786 个动宾短语,其中,2 413 个动宾短语修正、合并成为最终的软件功能特征.在软件功能特征视图中,对象与功能特征之间建立了 11 176 个“has_function”关联,对象之间基于继承关系建立了 1 081 个“inherits”关联,功能特征之间基于来源于相同 Stack Overflow 讨论帖建立了 1 637 个“from_same_post”关联.

Table 3 Statistics for functional feature mining

表 3 功能特征挖掘总体统计情况

软件项目	基本功能特征			复合功能特征		功能特征关联关系		
	种子概念	扩展后概念	基本功能特征	动宾短语	复合功能特征	has_function	inherits	From_same_post
Apache POI	1 320	2 072	8 763	12 786	2 413	11 176	1 081	1 637
Eclipse JDT	1 744	1 998	9 265	2 805	899	10 164	947	582
JFreeChart	609	1 056	4 042	6 913	1 067	5 109	546	725

为了进一步分析本文的软件功能特征挖掘结果,我们将挖掘得到的功能特征与软件项目的官方文档进行

了对比分析.对比评价标准见表 4.其中,“功能特征”指的是本文方法挖掘出的软件功能描述;“功能条目”指的是软件项目功能描述文档中的软件功能描述;“代码片段”指的是功能特征来源于的 Stack Overflow 讨论帖中所包含的代码信息;“代码示例”指的是软件项目功能描述文档中的代码信息.实验中我们逐一检视这些软件功能描述,采用人工评价的方式判断该软件功能描述是否出现在本文方法的挖掘结果中.具体地,我们邀请了两位熟悉实验中使用的开源项目的开发者,分别依次判断是否符合 T1~T4 的覆盖情况,把功能特征首次符合的覆盖情况作为该功能特征的覆盖类型.每位开发者独立进行判断,全部完成后对意见不一致的功能特征统一进行评判.

Table 4 The coverage evaluation criteria in our experiments

表 4 挖掘结果与官方文档中功能特征的评价策略

类型	覆盖情况
T1	功能特征与功能条目具有相同的名称
T2	功能特征与功能条目具有相同的含义,但描述方式不同
T3	代码片段中的注释与功能条目具有相同的名称或含义
T4	代码片段与代码示例调用相同的 API

表 5 描述了针对上述两个项目进行对比分析的结果.其中,Apache POI 软件官方文档中包含 Busy Developers’ Guide to Features 等 46 个功能条目.从表中可以看到,该项目评价为功能特征与功能条目具有相同的名称(T1)和功能特征与功能条目具有相同的含义但描述方式不同(T2)的功能特征共计占比 69.57%,4 种覆盖类型总共占比 95.65%.而在 Eclipse JDT 的 19 个功能条目中,评价为功能特征与功能条目具有相同的名称(T1)和功能特征与功能条目具有相同的含义但描述方式不同(T2)的功能特征共计占比 52.63%,4 种覆盖类型总共占比也达到了 94.74%.综合两个软件项目可以得出,本文方法对功能描述文档中列举的常用软件功能达到了 95.38%的覆盖率.

Table 5 Results for coverage evaluation

表 5 挖掘结果对官方功能文档的覆盖率情况

类型	Apache POI 功能特征(46)		Eclipse JDT 功能特征(19)	
	相符的功能特征数	占比(%)	相符的功能特征数	占比(%)
T1	27	58.70	8	42.11
T2	5	10.87	2	10.53
T3	1	2.17	1	5.26
T4	11	23.91	7	36.84
总计	44	95.65	18	94.74

在 Apache POI 的功能描述文档中,有两个功能条目没有出现在本文方法挖掘出的软件功能特征列表中.经过观察、分析后发现,其中一个功能条目在 Stack Overflow 问答网站上没有相关的讨论记录,因此无法挖掘出来;另一个功能条目的相关信息仅在非接受回答(acc_answer)中出现了 1 次,非接受回答相当于没有受到讨论帖作者的认可,根据本文对讨论帖中内容的质量和正确性的考量,不考虑非接受回答中的内容,因此也无法挖掘出来.而在 Eclipse JDT 的功能描述文档中,仅有一个功能条目没有出现在本文方法挖掘出的软件功能特征列表中.调研结果表明,由于该功能条目在 Stack Overflow 问答网站上也没有相关的讨论记录,因此无法挖掘出来.

3.3 RQ2:挖掘算法的准确率分析

本文实验从 Apache POI、Eclipse JDT 和 JFreeChart 这 3 个软件项目的讨论帖中分别提取了 13 993、2 057 和 5 968 个语句,要逐一分析每个语句挖掘结果是不现实的.为此,我们从各个项目的讨论帖中分别随机抽取 300 个语句,由熟悉这 3 个软件项目的开发人员人工标注每个语句中出现的软件功能特征,作为标准数据对照集.为了避免人工选取动宾短语带来的误差,我们实现用 spaCy 工具抽取了句子中的动宾短语,标注时仅需要选择符合标准的动宾短语,而非人工截取句中的动宾短语.为避免标注错误,只有同时被 3 位人员标注为功能特征的动宾短语才被保留.标准数据集中软件功能特征的分布情况如图 4 所示.各个软件项目的语句中的功能特征分布情况大致相同:大部分实验语句中包含 0 个软件功能特征,即语句中没有提及软件功能相关的信息;少部分(几十个)语句中提及 1 个软件功能特征;极少量(不足 10 个)实验语句中提及 2 个或 3 个软件功能特征;没有语句提及

4 个及以上软件功能特征.

之后,我们请构造标准对照数据集的 3 位开发人员浏览本文方法生成的功能特征列表,并对其中每个条目的准确性进行评判.对于实验数据中的每个语句,如果人工标注的软件功能特征集合与本文方法挖掘出的软件功能特征集合完全一致,则认为本文方法对该语句的挖掘结果是正确的.实验结果见表 6.

从表 6 可以看到,在人工标注的标准数据集中,实验人员平均为每个软件项目的 300 个语句标注了 76.00 个功能特征.在本文方法的语句挖掘结果中,为 Apache POI 软件项目中的 276 个语句正确挖掘出了对应的功能特征,为 Eclipse JDT 和 JFreeChart 软件项目中的 284 个语句正确挖掘出了对应的功能特征,平均的挖掘正确率达到了 93.78%.在软件功能特征挖掘结果中,为 Eclipse JDT 软件项目挖掘 58(54+4)个功能特征,其中,54 个与人工标注相符,4 个没有出现在人工标注的结果中,准确率为 93.10%,同时工具遗漏了 12 条人工标注的功能特征,挖掘结果的召回率为 81.82%.平均情况下,每个软件项目有 62.33 个功能特征挖掘正确,有 13.67 个软件功能特征被遗漏,有 5 个软件功能特征提取错误,功能特征挖掘结果的召回率和准确率分别达到了 81.35%和 92.57%.

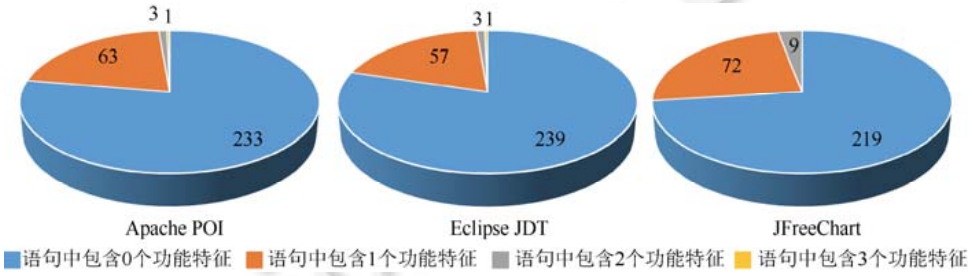

Fig.4 Distribution of functional features in sentences of standard dataset

图 4 标准数据集的语句中软件功能特征分布情况

Table 6 Accuracy scores of the generated functional feature list

表 6 本文方法挖掘结果与标准数据集相比的准确率评估结果

软件项目	人工标注		语句挖掘结果		功能特征挖掘结果				
	语句数	功能特征数	正确	正确率(%)	正确	遗漏	错误	召回率(%)	准确率(%)
Apache POI	300	72	276	92.00	52	20	4	72.22	92.86
Eclipse JDT	300	66	284	94.67	54	12	4	81.82	93.10
JFreeChart	300	90	284	94.67	81	9	7	90	92.05
平均	300.00	76.00	281.33	93.78	62.33	13.67	5.00	81.35	92.57

3.4 RQ3:挖掘方法的对比分析

本文综合利用软件项目源代码和 Stack Overflow 讨论帖来挖掘软件功能特征,在设计挖掘方案时对两部分数据分别进行了考虑:对于源代码信息,本文从源代码中提取 API 作为基本软件功能特征名称的来源,并从源代码中挖掘软件概念以辅助筛选软件功能特征名称;对于 Stack Overflow 信息,本文利用自然语言处理工具对 Stack Overflow 讨论帖中的文本信息进行句法分析,并利用启发式规则筛选出软件功能特征名称.为了比较验证本文方法中这两方面数据源的必要性,实验中我们分别实现了以单一数据源为输入的功能特征挖掘方法进行比较.同时,我们对比了两个现有方法:TaskNav[13]和 APITasks[17].TaskNav 方法基于自然语言处理中的实体识别技术选取软件文档中的概念词汇,并基于人工制定的动词白名单对文档中出现的动宾短语进行筛选,从而返回与本文功能特征形式一致的(动宾短语形式)的软件开发任务.APITasks 的场景与本文一致,即从用户交流记录中抽取编程任务相关的动宾短语,该方法使用 3 种过滤规则(短语的结构、上下文和停用词)剔除低质量的动宾短语.对比时,我们获取了 TaskNav 的网页工具(http://task-phrases.herokuapp.com)和 APITasks 的项目源码,将数据集中的句子依次输入给这两种工具,将返回结果和本文方法输出一起评判.综上,本节对比了 TaskNav 和 APITasks 两种工具,并对本文方法的两个变种(“只考虑源代码信息”和“只考虑 Stack Overflow 信息”)进行了实验,判断实验语句集合中可被正确挖掘的句子数量,考虑到不同工具在动宾短语的形式定义上有细微区别,在判

定时我们使用 spaCy 工具选取动宾短语中的核心动词和核心宾语成分作为比较对象,只有词根化后动词、宾语完全一致的动宾短语才被认定为正确.最终对比结果见表 7.

从表 7 可以看出,TaskNav 方法抽取正确的句子比例较低,一个重要原因是 TaskNav 基于人工指定的动词白名单对动宾短语进行筛选,而不同项目之间的领域动词存在差异(如 Eclipse-JDT 中的高频动词“visit”没有被 TaskNav 纳入白名单).APITasks 工具虽然制定了 3 种过滤策略对低质量动宾短语进行过滤,但起到主要过滤作用的仍然是简单的停用词匹配.与 TaskNav 不同,APITasks 给出了动词的过滤黑名单,但同样面临着不同领域的动词集合存在差异这一问题.“只考虑代码信息”和“只考虑 Stack Overflow 信息”这两种方法的平均准确率都刚好达到 80%,而本文将这两种方法结合起来挖掘的准确率接近 94%.因此,本文提出的挖掘方法总体而言效果显著,两种数据源的融合/挖掘策略也是必要的.

Table 7 Precision scores of approach comparison

表 7 本文方法比较的准确率评估结果

软件项目	TaskNav	APITasks	本文方法	只考虑源代码信息	只考虑 Stack Overflow 文档信息
Apache POI	69.00	82.00	92.00	80.67	80.67
Eclipse JDT	56.00	85.67	94.67	80.00	83.33
JFreeChart	70.00	79.00	94.67	80.00	78.00
平均	65.00	82.22	93.78	80.22	80.67

本文把从软件项目源代码中提取出的代码元素中的类名作为种子概念,基于规则进行迭代扩充,得到软件概念集合.本实验通过对比软件概念迭代扩充次数对挖掘结果准确率的影响来评估软件概念迭代挖掘算法的有效性.实验结果见表 8.

从实验结果可以看出,软件概念迭代扩充 1 次和迭代扩充 2 次对软件功能特征准确率的影响没有差别,这是因为迭代扩充过程的终止条件是没有新增的软件概念,因此,最后一次扩充前后的软件概念集合是相同的,因而对软件功能特征准确率的影响也是相同的.迭代扩充 2 次后即满足迭代扩充的终止条件,也就是说,在对软件概念进行 1 次扩充后即获得本文方法最终挖掘出的软件概念集合.这是因为,本文将软件项目源代码中定义的类的名称都作为种子概念,以此为软件概念挖掘算法的起点.经历第 1 次扩充规则后,新增的软件概念属于以下两种情况:(1) 新增的软件概念所对应的类是在软件项目源代码中定义的.由于这些类的名称在算法初始时就存在于种子概念集合中,因此后续的扩充不会因为这些类对软件概念集合产生影响.(2) 新增的软件概念所对应的类不是在软件项目源代码中定义的.由于本文方法是依据源代码中描述的类的继承关系和类中的方法来扩充软件概念的,因此后续的扩充不会从这些类中挖掘出新的软件概念.

Table 8 Influence of iterative extraction algorithm on precision of functional feature (%)

表 8 本文软件概念迭代扩充算法对软件功能特征准确率的影响(%)

软件项目	Apache POI	Eclipse JDT	JFreeChart	平均
未扩充	89.33	88.67	84.67	87.56
迭代扩充 1 次	92.00	94.67	94.67	93.78
迭代扩充 2 次	92.00	94.67	94.67	93.78

从挖掘结果来看,按照本文规则将种子概念集合迭代扩充得到软件概念集合,以辅助软件功能特征的筛选过程,可以将软件功能特征挖掘结果的准确率由 87.56%提升为 93.78%.因此,本文设计的软件概念迭代扩充算法是有效的,对软件功能特征挖掘算法的辅助作用是显著的.

3.5 有效性分析

本文方法的思路是从软件项目源码中提取领域词汇,进而到文档交流记录中过滤出动宾短语形式的软件功能特征.需要指出的是,输入的数据质量可能会影响到本文方法的效果.

首先,本文依赖源码的标识符具有较丰富的语义信息和规范的命名风格.本文针对 Java 语言常见的驼峰命名进行了切词和概念抽取,该过程可以方便地扩展到其他变量命名风格(如下划线分隔).虽然本文方法考虑了缩写词等命名习惯对概念抽取的影响,但仍会有一些概念匹配错误的情况发生,如拼写错误和单词内部的缩写

(context 缩写成 ctx).因此,除了本文提到的启发式匹配规则外,应用一些基于共现的统计方法[16]有可能会降低本文方法对命名风格的敏感性.

其次,文档交流记录的丰富程度会影响到最终挖掘的功能特征数量.本文使用 Stack Overflow 作为文档输入来源,如果一个项目在该网站上没有足够的讨论记录,本文方法就不能生成高质量的软件功能特征.但需要指出的是,本文方法并没有基于 Stack Overflow 对输入文档的格式作特定假设,因此可以较方便地扩展到其他类型的交流记录(项目特定的论坛、邮件列表等).本文使用 Stack Overflow 仅仅因为其数据比较规整,包含大量关于不同开源项目的讨论.同时,由于不需要训练学习过程,本文方法的准确率并不受文档规模的影响.

4 相关工作

本文的相关工作主要从各类软件文档中抽取软件的功能描述信息.代表性的工作是 Treude 等人[11,13]将开发任务定义为描述特定编程操作的动宾短语,并提出了一种基于开发任务索引和浏览软件官方文档的方法.基于该方法的工具 TaskNav 可以帮助开发人员更高效地浏览文档,定位自身复用需求.该团队后续开发了 NLP2Code 工具[18]以 Eclipse 插件形式在 IDE 中推荐相关的开发任务和示例代码.朱子骁等人[17]提出了基于 Stack Overflow 数据的软件功能特征挖掘组织方法.该方法没有考虑源代码信息,采用自然语言处理与频繁子图挖掘的方式获得软件功能特征,取得了较好的覆盖率,但准确率较低.Panichella 等人[19]以开发者交流渠道为数据来源,为 API 类或方法抽取描述信息;Rastkar 等人[20]提出了一种对邮件、缺陷报告等软件交流数据生成文本摘要的技术;Wang 等人[21]用监督学习的方法得到了文档中描述领域概念的句法模式,进而自动地从软件功能文档中抽取领域概念;Wong 等人[15]开发的 AutoComment 工具从 Stack Overflow 上匹配可以作为代码注释的语句.考虑到论坛中存在大量与软件功能无关的自然语言,该工具采用启发式规则对语句进行过滤,并采用动宾短语的形式进行总结;类似的工作还包括 Silva 等人[22]开发的 CROKAGE 工具,Jiang 等人[23]采用无监督学习方法开发的 API 教程推荐工具 FRAPT,Treude 等人[24]提出的一种挖掘阐述 API 使用关键句(insight)的有监督学习方法 SISE 等等.可以看出,从各类软件数据中获取软件描述信息的工作已经得到了很大的关注,但这些工作在挖掘软件功能描述方面仍存在准确率的问题,且部分工作生成的代码描述未必是对软件功能的描述,只是代码相似的文本.本文注重从软件功能角度出发,综合代码和 Stack Overflow 两种数据源进行挖掘,获得了较好的结果.

此外,一些工作关注代码注释[16]或代码总结[25]的自动生成,这些注释通常同样包含软件的功能描述.此类工作具体可分为基于模板的注释生成和基于机器翻译的注释生成两种策略.基于模板的代码注释生成方法需要首先按照预定义代码模式抽取关键代码信息,包括抽取关键字、抽取关键语句、抽取代码关键结构、抽取代码类型信息和抽取上下文信息等.代表性的工作包括:Haiduc 等人提出了从代码中选择出的 n 个词以及代码结构信息作为代码注释的方法[26],其后他们又利用 VSM 和 LSI 抽取代码中的重要部分作为代码的总结[27];SWUM 方法则通过一些规则识别代码中的关键语句,为一个函数生成功能性描述[28,29];Moreno 等人提出了基于特定的生成模板为面向对象语言中的类产生注释的方法[30];McBurney 等人则提出了一种侧重上下文的注释生成方法 NLG[31],等等.基于机器翻译的注释生成方法将软件代码看作字符串序列,采用双语翻译的机器学习方法或模型生成自然语言的代码注释.选择代码中的何种信息用于表示代码是该问题的关键.一种较为简单的方法是:将代码视为单词序列,并用代码中出现的单词序列用于表示代码并作为模型的输入.代表性的工作是 Iyer 等人提出的基于 LSTM 模型和 attention 机制的注释自动生成方法 CODE-NN[32].然而这种处理方法必然会丢失代码中的结构信息.为此,后续工作除了利用代码本身单词的信息以外,同时还对代码进行解析,抽取代码的结构信息.这些信息大多数是基于代码的抽象语法树获取,代表性的工作包括 Hu 等人提出的自动注释方法 DeepCom[33]、Alon 等人提出的 Code2seq[34]等等.尽管注释生成工具同样可以生成自然语言形式的功能描述,但通常针对一个类、方法(含 API)或一段逻辑完整的代码,从用户场景上与本文针对软件项目的功能特征挖掘工作有所不同.在刚接触一个软件项目库时,获得一段高质量但待解释的示例代码并不容易,本文工作能够帮助新用户快速浏览定位到所需功能.

5 总结与展望

本文提出了一种融合代码与文档的软件功能特征挖掘方法,挖掘结果可以帮助开发人员快速了解待复用软件项目的功能特征,提高软件复用的效率.该方法以动宾短语形式描述一项软件功能特征,通过迭代挖掘源代码抽取软件概念,辅助从源代码和 Stack Overflow 问答文档中挖掘软件功能特征,并构建层次化的软件功能特征视图.我们在大量软件项目数据上进行了实验.实验结果表明,本文方法获取的软件功能特征可以覆盖官方文档中约 95%的软件常用功能,挖掘结果中语句和功能特征的准确率分别达到了 93.78%和 92.57%.

我们下一步的工作计划包括:研究如何改进本文方法中的动宾短语提取方法,提高从自然语言文本中提取功能特征动宾短语的准确率;增强本文工具综合挖掘多种类型的软件数据的能力,例如从开发者/用户邮件列表、缺陷追踪报告中进一步提取相关的功能特征;在实验上,将本文方法和工具应用到更多的软件项目和代码库中,通过大规模数据进行实例验证.

References:

[1] Mili H, Mili F, Mili A. Reusing software: Issues and research directions. IEEE Trans. on Software Engineering, 1995,21(6): 528–562. [doi: 10.1109/32.391379]

[2] Yang FQ, Mei H, Li K. Software reuse and software component technology. Acta Electronica Sinica, 1999,27(2):68–75,51 (in Chinese with English abstract).

[3] Theotokis SA, Spinellis D, Kechagia M, *et al*. Open source software: A survey from 10,000 feet. Foundations and Trendső in Technology, Information and Operations Management, 2011,4(3–4):187–347. [doi: 10.1561/0200000026]

[4] Robillard MP, Deline R. A field study of API learning obstacles. Empirical Software Engineering, 2011,16(6):703–732. [doi: 10.1007/s10664-010-9150-8]

[5] Barthélémy D, Robillard MP. Recovering traceability links between an API and its learning resources. In: Proc. of the 34th Int'l Conf. on Software Engineering (ICSE). IEEE, 2012. 47–57. [doi: 10.1109/ICSE.2012.6227207]

[6] Shen Q, Xie B, Zou YZ, Zhu ZX, Wu SJ. Nli2code: Reusing libraries with natural language interface. In: Proc. of the 18th Int'l Conf. on Software and Systems Reuse (ICSR). Springer-Verlag, 2019. 168–184. [doi: 10.1007/978-3-030-22888-0_12]

[7] Ding W, Liang P, Tang A, *et al*. Knowledge-based approaches in software documentation: A systematic literature review. In: Information and Software Technology, 2014. 545–567. [doi: 10.1016/j.infsof.2014.01.008]

[8] Zhi JJ, Yusifoğlu VG, Sun B, *et al*. Cost, benefits and quality of software development documentation: A systematic mapping. Journal of Systems and Software, 2015, 175–198. [doi: 10.1016/j.jss.2014.09.042]

[9] Garousi G, Yusifoğlu VG, Ruhe G, *et al*. Usage and usefulness of technical software documentation: An industrial case study. In: Information and Software Technology. 2015. 664–682. [doi: 10.1016/j.infsof.2014.08.003]

[10] 中华人民共和国国家质量监督检验检疫总局,中国国家标准化管理委员会.GB/T 11457 信息技术软件工程术语.2006.

[11] Treude C, Mathieu S, Klocke M, Robillard MP. TaskNav: Task-based navigation of software documentation. In: Proc. of the 37th Int'l Conf. on Software Engineering (ICSE). IEEE, 2015. 649–652. [doi: 10.1109/ICSE.2015.214]

[12] Zhao W, Zhang L, Mei H, Sun JS. A functional requirement based hierarchical agglomerative approach to program cluster. Ruan Jian Xue Bao/Journal of Software, 2006,17(8):1661–1668 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825 17/1661.htm

[13] Treude C, Robillard MP, Barthélémy D. Extracting development tasks to navigate software documentation. IEEE Trans. on Software Engineering, 2014,41(6):565–581. [doi: 10.1109/TSE.2014.2387172]

[14] Sun J, Xing Z, Chu R, Bai H, Wang J, Peng X. Know-how in programming tasks: From textual tutorials to task-oriented knowledge graph. In: Proc. of the 35rd Int'l Conf. on Software Maintenance and Evolution (ICSME). IEEE, 2019. 257–268. [doi: 10.1109/ICSME.2019.00039]

[15] Wong E, Yang JQ, Tan L. AutoComment: Mining question and answer sites for automatic comment generation. In: Proc. of the 28th Int'l Conf. on Automated Software Engineering (ASE). IEEE, 2013. 562–567. [doi: 10.1109/ASE.2013.6693113]

[16] Hu X, Li G, Xia X, Lo D, Jin Z. Deep code comment generation. In: Proc. of the 26th Int'l Conf. on Program Comprehension. ACM, 2018. 200–210. [doi: 10.1145/3196321.3196334]

[17] Zhu ZX, Zou YZ, Hua CY, Shen Q, Zhao JF. Mining and organizing software functional features based on StackOverflow data. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2210−2225 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5533.htm [doi: 10.13328/j.cnki.jos.005533]

[18] Campbell BA, Treude C. NLP2Code: Code snippet content assist via natural language tasks. In: Proc. of the 33rd Int'l Conf. on Software Maintenance and Evolution (ICSME). IEEE, 2017. 628–632. [doi: 10.1109/ICSME.2017.56]

[19] Panichella S, Jairo A, Massimiliano DP, Andrian M, Gerardo C. Mining source code descriptions from developer communications. In: Proc. of the 20st Int'l Conf. on Program Comprehension (ICPC). IEEE, 2012. 63–72. [doi: 10.1109/ICPC.2012.6240510]

[20] Sarah R, Murphy GC, Murray G. Summarizing software artifacts: A case study of bug reports. In: Proc. of the 32nd Int'l Conf. on Software Engineering (ICSE). ACM, 2010. 505–514. [doi: 10.1145/1806799.1806872]

[21] Wang C, Peng X, Liu MW, Xing ZC, Bai XF, Xie B, Wang T. A learning-based approach for automatic construction of domain glossary from source code and documentation. In: Proc. of the Foundations of Software Engineering (FSE). 2019. 97–108. [doi: 10.1145/3338906.3338963]

[22] Silva R, Roy C, Rahman M, Schneider K, Paixao K, Maia M. Recommending comprehensive solutions for programming tasks by mining crowd knowledge. In: Proc. of the 27th Int'l Conf. on Program Comprehension (ICPC). IEEE, 2019. 358–368. [doi: 10.1109/ICPC.2019.00054]

[23] Jiang H, Zhang JX, Ren ZL, Zhang T. An unsupervised approach for discovering relevant tutorial fragments for APIs. In: Proc. of the 39th Int'l Conf. on Software Engineering (ICSE). IEEE, 2017. 38–48. [doi: 10.1109/ICSE.2017.12]

[24] Treude C, Robillard MP. Augmenting API documentation with insights from stack overflow. In: Proc. of the 38th Int'l Conf. on Software Engineering (ICSE). IEEE, 2016. 392–403. [doi: 10.1145/2884781.2884800]

[25] LeClair A, Jiang S, McMillan C. A neural model for generating natural language summaries of program subroutines. In: Proc. of the 41st Int'l Conf. on Software Engineering (ICSE). IEEE, 2019. 795–806. [doi: 10.1109/ICSE.2019.00087]

[26] Haiduc S, Aponte J, Marcus A. Supporting program comprehension with source code summarization. In: Proc. of the 32nd Int'l Conf. on Software Engineering (ICSE). ACM, 2010. 223–226. [doi: 10.1145/1810295.1810335]

[27] Haiduc S, Aponte J, Moreno L, Marcus A. On the use of automated text summarization techniques for summarizing source code. In: Proc. of the 17th Working Conf. on Reverse Engineering (WCRE). IEEE, 2010. 35–44. [doi: 10.1109/WCRE.2010.13]

[28] Sridhara G, Hill E, Muppaneni D, Pollock L, Shanker VK. Towards automatically generating summary comments for Java methods. In: Proc. of the 25th Int'l Conf. on Automated software engineering (ASE). ACM, 2010. 43–52. [doi: 10.1145/1858996.1859006]

[29] Sridhara G, Pollock L, Shanker VK. Automatically detecting and describing high level actions within methods. In: Proc. of the 33rd Int'l Conf. on Software Engineering (ICSE). ACM, 2011. 101–110. [doi: 10.1145/1985793.1985808]

[30] Moreno L, Aponte J, Sridhara G, Marcus A, Pollock L, Shanker VK. Automatic generation of natural language summaries for Java classes. In: Proc. of the 21st Int'l Conf. on Program Comprehension (ICPC). IEEE, 2013. 23–32. [doi: 10.1109/ICPC.2013.6613830]

[31] McBurney PW, McMillan C. Automatic documentation generation via source code summarization of method context. In: Proc. of the 22nd Int'l Conf. on Program Comprehension (ICPC). ACM, 2014. 279–290. [doi: 10.1145/2597008.2597149]

[32] Iyer S, Konstas I, Cheung A, Zettlemoyer L. Summarizing source code using a neural attention model. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (ACL). 2016,1:2073–2083. [doi: 10.18653/v1/p16-1195]

[33] Hu X, Li G, Xia X, Lo D, Jin Z, Deep code comment generation with hybrid lexical and syntactical information. Empirical Software Engineering, 2020,25(3):2179–2217.

[34] Alon U, Levy O, Yahav E. Code2seq: Generating sequences from structured representations of code. In: Proc. of the 7th Int'l Conf. on Learning Representations (ICLR). OpenReview.net, 2019.

附中文参考文献:

[2] 杨芙清,梅宏,李克勤.软件复用与软件构件技术.电子学报,1999,27(2):68–75,51.

[10] 中华人民共和国国家质量监督检验检疫总局,中国国家标准化管理委员会.GB/T 11457 信息技术软件工程术语.2006.

[12] 赵伟,张路,梅宏,孙家骕.一种基于功能需求层次凝聚的程序聚类方法.软件学报,2006,17(8):1661–1668. http://www.jos.org.cn/1000-9825/17/1661.htm

[17] 朱子骁,邹艳珍,华晨彦,沈琦,赵俊峰.基于 StackOverflow 数据的软件功能特征挖掘组织方法.软件学报,2018,29(8):2210–2225. http://www.jos.org.cn/1000-9825/5533.htm [doi: 10.13328/j.cnki.jos.005533]

沈琦(1995－),男,博士生,主要研究领域为软件工程,软件复用,代码自动生成.

钱莹(1994－),女,硕士,主要研究领域为软件工程,软件复用.

邹艳珍(1976－),女,博士,副教授,CCF 专业会员,主要研究领域为软件工程,软件复用,知识图谱,智能软件开发.

伍仕骏(1998－),男,博士生,主要研究领域为软件工程,软件复用.

谢冰(1970－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,形式化方法,软件复用,智能软件开发.