DOI: 10.3969/j.issn.0255-8297.2020.05.003

• 特邀论文 •

基于构建软件功能描述的可复用软件检索方法

伏广宇[1,2]， 李传艺[1,2]， 葛季栋[1,2]， 骆 斌[1,2]
1. 南京大学 计算机软件新技术国家重点实验室，南京 210046
2. 南京大学 软件学院，南京 210093

摘 要： 软件复用技术能够有效降低新软件产品开发的时间、人力和成本. 在软件复用中，基于待开发软件的基本描述与简单需求，如何衡量已有软件的可复用性并对大量已有软件进行快速、自动的可复用性评估，已成为首要解决的问题. 目前已有较多评价软件产品或软件项目相似度的研究工作，但相似性并不等于可复用性. 因此，该文通过调研软件产品可复用性的相关研究，定义了一套适用于开源软件仓库中软件项目的可复用性评价指标，并设计了基于待开发软件项目的基本需求快速查询可复用软件项目的算法，实现了可复用软件项目检索系统. 通过实验及专家对检索结果的评价，验证了所提描述方法的高效性和可用性.

关键词： 软件复用；软件需求；软件相似性；软件项目推荐；软件可复用性；开源软件仓库
中图分类号： TP311.5 **文章编号：** 0255-8297(2020)05-0682-13

Retrieving Reusable Software by Constructing Functional Descriptions

FU Guangyu[1,2], LI Chuanyi[1,2], GE Jidong[1,2], LUO Bin[1,2]
1. State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China
2. Software Institute, Nanjing University, Nanjing 210093, China

Abstract: Through software reuse technology, reusing existing software components and modules can effectively reduce the time, labor and costs of new software product development. In software reuse, how to measure and evaluate the reusability of existing software is the first problem to be solved. Although there are a lot of researches assessing the similarities, it is not equal to the reusability. Therefore, this paper defines a set of assessment indexes which is applicable to the reusability of software projects in open source software repository, then designs an algorithm to quickly query reusable software projects based on the basic requirements of the software to be developed, and finally completes the retrieval system of the reusable software project. Experimental results and expert evaluation of the retrieval results verify the efficiency and usability of the method.

收稿日期： 2020-06-14
基金项目： 国家自然科学基金（No.61802167）；南京大学中央高校基本科研业务费资助
通信作者： 李传艺，博士，助理研究员，研究方向为软件需求工程. E-mail：lcy@nju.edu.cn

Keywords: software reuse, software requirements, software similarity, software project recommendation, software reusability, open-source repository

随着互联网的飞速发展和软件规模的不断扩大，软件复用已成为降低开发成本、节约开发时间的有效途径. 越来越多的开发者参与到开源软件的开发中来，积累了大量可复用的代码、组件、服务，为开发者能够快速构建新软件提供了有利条件. 大量开源软件社区也随之兴起（如 Github、SourceForge、OSCHINA、OpenHub 等）. 这些开源软件社区为开发者提供了开源软件托管的平台，同时支持开源软件检索，使用户能够找到相关的开源软件进行复用.

在现有的开源软件社区中，大部分只提供了根据软件仓库名称、标签、分类、软件描述等信息进行检索的方法. 开源软件社区中存在着大量优秀的开源软件但缺少详细的描述，不能将软件的所有功能体现出来，导致用户检索困难. 然而，技术问答社区（如 Stack Overflow）中存在关于开源软件的问答描述，为软件提供了更加详细的信息，可以成为可复用软件检索的辅助信息来源.

已经有很多关于可复用软件检索的研究，如 Bajracharya 等[1]实现了 Sourcerer 系统，从代码角度进行分析，提取出细粒度信息存储在关系模型中用于进行搜索，效果远优于基于关键词的搜索. Gu 等在文献 [2] 中提出了一种新的带有层次结构的软件标签结构，优化了基于标签的软件搜索方式. Girardi 等[3]构建了一个名为 ROSE 的方法，通过对软件描述文本中的动词与名词进行抽取，建立索引为用户提供软件检索.

然而，这些研究都没有解决用户输入与开源软件仓库索引方式之间的不对等问题. 现有研究中要么使用软件代码（如 SCRUPLE 工具[4]），要么使用软件简介（如 RepoPal 系统[5]）对软件进行索引，而输入则为用户表达的非正式的软件需求描述. 同时，开源软件社区中存在着大量优秀开源软件的标签、描述不全的问题，用户不能针对软件的功能特征进行检索. 因此，本文提出了一种通过开源软件论坛或问答网站构建软件功能描述，以匹配用户输入的方法，用于满足用户针对软件功能进行检索的需求，具体包括：

1）通过将开源软件与相关论坛结合，从论坛上提取出软件所具有的功能特征；

2）通过对用户描述的软件需求文本处理，快速提取出用户所需的软件功能；

3）用 BERT(bidirectional encoder representation from transformers)[6]预训练模型将软件功能特征与用户所需软件功能转换为句向量进行相似度计算并推荐.

1　开源软件项目的复用

1.1　开源软件项目可复用性定义

软件复用是用现有的软件或者软件知识来构建新的软件[7]. 在软件开发过程中，软件复用有几个不同的层次，如代码复用、设计复用、需求复用等. 代码级复用[8]包括函数复用、模块复用、库复用、完整的软件产品复用等；设计级复用[9]包括模式复用、设计图表复用、架构复用、框架复用等；需求复用[10]则包括非正式需求描述复用与需求规格说明复用等. 无论何种软件复用形式，其目的都是提高软件质量和生产率，这也成为许多科学家与管理学家关注的焦点. 软件的可复用性是衡量软件是否能被复用的重要指标，可以通过以下几个方面进行衡量：

1）软件功能的复用

在设计初期，许多软件的存在形式具有高度的可复用性. 开发者可以将现有项目中的源代码或组件直接或简单修改后应用到新的软件中. 例如在 Java 开发中，开发者通过引入 Java Archive（Jar）文件，能够快速、高效地使用拓展功能. 面向对象程序设计的出现进一步推动了

软件复用的发展. 通过对软件中功能与实体的抽象，通过类、方法、对象将软件的功能与角色进行封装，一次编写即可随时被调用，大大降低了开发成本.

2）软件体系结构的复用

随着领域工程、领域分析的出现，软件的体系结构成为了复用软件的重要考虑因素. Arango 等[11]提出了一种在受限领域中构建重用软件的理论. 在特定领域中，通过对用户反馈与成功因素等积累，能够大幅度提高软件开发的能力与效率. 因此，通用模版也成为了软件复用重要资源.

3）开发过程重用

过程重用是最抽象、最高级的重用方法，与对特定代码片段甚至是软件体系结构的重用无关. 这其中包括项目团队的开发工具、开发周期、协作模式、开发中所产生的经验等. Barreto 等[12]提出了一种软件流程的定义方法来获取软件开发过程中可复用的元素，如团队组织、进度安排等.

本文主要研究检索满足功能可复用的开源软件项目，而如何对检索到的可复用软件项目进行复用不是本文研究内容. 本文使用自然语言表达软件的功能特征.

在软件复用时，通常会面临道德与法律问题. 开源软件快速增长的同时也带动了开源社区与开源协议的进步与发展. 为了避免道德问题与法律纠纷并保护开源软件作者的权益，开源软件协议也是评估开源软件复用性的重要因素之一[13]. 常见的开源协议有 MIT License（MIT）、GNU General Public License v3.0（GLP）、Apache Licence 2.0（Apache-2.0）、Berkeley Software Distribution（BSD）等. Lerner 在文献 [14] 指出，不同的开源协议对开源软件的用途、规范的限制不同，所以在选择开源软件进行复用时，需要考虑开源软件所遵守的协议. 本文描述的方法只限定使用于符合开源软件复用协议的相关软件.

1.2 本文可复用软件检索目标

本文设定的可复用软件检索目标为：检索基于软件功能可复用的相似开源软件. 通过对开源软件的功能特征与需求文档中的功能特征进行匹配，为用户推荐具有可被直接复用功能的开源软件，或具有相似功能经过修改可被复用的开源软件.

现有开源软件仓库对软件本身功能性描述并不全面，主要是对软件代码的管理和软件迭代更新过程的管理. 为了实现基于可复用功能的开源软件检索，首先需要明确每一个开源软件的功能性需求，作为判断是否符合检索要求的基础. 虽然软件仓库没有详细描述软件的功能性需求，在一些用户论坛和问答系统中，软件用户会对相关软件的功能开展讨论，从用户发言中可以抽取出软件的功能性描述. 因此，本文提出一种从用户论坛或问答系统中抽取软件功能描述的方法，用于构建软件功能索引.

从用户描述语言中提取软件功能描述的另一个优点是，其描述语言与用户输入的检索条件保持一致. 一般而言，普通用户在描述软件功能时不会使用专业术语，更加偏向日常使用的自然语言描述，如果将这种检索输入与正式的软件需求规格说明中的功能描述进行匹配，即使描述的为相同的功能，其文本相似程度也非常低. 虽然可以通过语义编码将规格说明与用户描述映射到同一个语义空间，其效果不如在输入和索引中均使用用户描述语言，这样通过语义编码后，语义相近的更可能匹配成功.

本文描述的检索系统的输入为自然语言描述的软件功能性需求，其特点是非正式的软件功能性描述，与写入软件需求规格说明的叙述存在一定差别. 为根据输入的功能描述检索功能相似的软件项目，将其与从相关论坛或问答社区中提取出的开源软件功能特征映射到同一个语义空间中，使用数值向量表示文本内容，例如通过现阶段流行的 BERT 神经网络语言模型

将两种文本转换为向量表示. 假设，v_i表示用户输入的功能描述的文本向量，v_j表示开源软件功能特征的文本向量，m为向量的维度，用余弦相似度公式计算两者相似性如下：

$$S_{<i,j>} = \cos\theta = \frac{\sum_{p=1}^{m}(v_{ip}v_{jp})}{\sqrt{\sum_{p=1}^{m}v_{ip}^2}\sqrt{\sum_{p=1}^{m}v_{jp}^2}} \tag{1}$$

同时，纳入本文描述系统检索范围的开源软件需遵守相关的开源许可，确保用户对开源软件的合法复用.

2　检索系统的详细叙述

本文提出了一种基于软件需求的开源软件检索方法. 图 1 展示了该方法的整体框架.

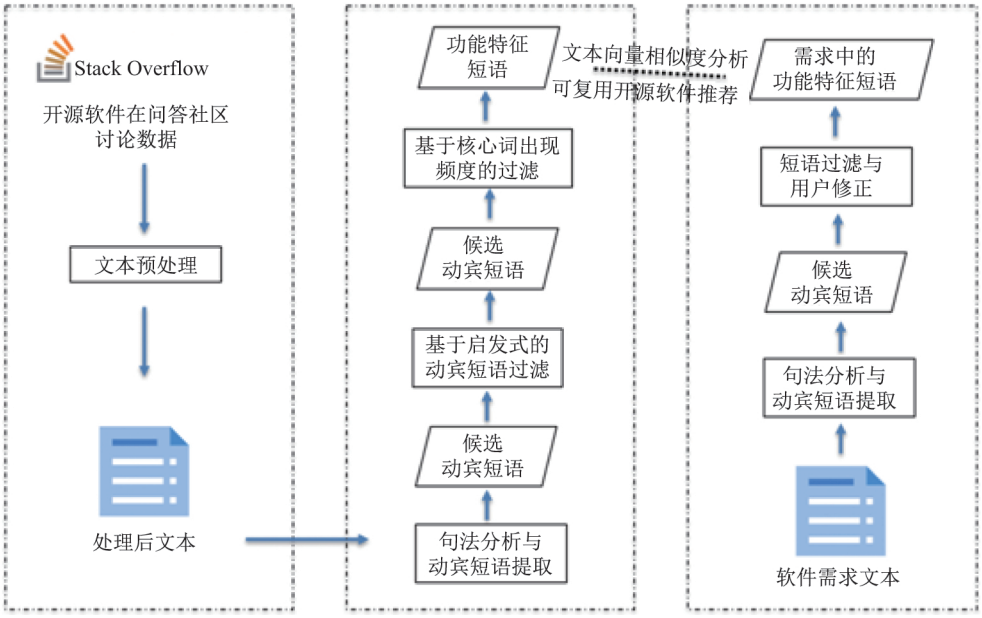

图 1　本文提出的方法整体框架图

Figure 1　Overall framework in our proposed approach

该方法的核心工作是为每一个开源软件构建功能特征描述，处理用户上传的功能需求描述，并计算开源软件功能特征与用户描述中所需功能的相似度. 开源软件是系统返回的结果单元. 该方法主要分为 4 个步骤：

步骤 1　对开源软件在论坛或问答社区中的文本讨论数据进行预处理（本文以开发者技术论坛 Stack Overflow 作为数据来源，基于该数据描述本文方法的具体流程并验证该方法的有效性），减少文本中代码、网页标签带来的负面影响；

步骤 2　通过句法分析与动宾短语提取从问答语句中筛选出动宾短语，作为开源软件功能特征的候选词，再通过启发式过滤规则和信息词出现的频率进行过滤，筛选出开源软件的功能特征短语；

步骤 3 对用户上传软件需求描述进行动宾短语提取与过滤，同时为用户提供自行筛选与修改动宾短语的接口，辅助用户提取出软件需求中所需的软件功能；

步骤 4 通过对提取出的开源软件功能与用户所需的软件功能进行文本向量转换，并计算相似度，最终根据相似度筛选出前 K 个结果推送给用户.

2.1 问答数据预处理

本文从 Stack Exchange Data Explorer 上获取数据源，根据开源软件的名称在 Stack Overflow 上对应的标签获取相关的问答信息. 如图 2 所示，Stack Overflow 中的问答记录以网页的形式进行展示，所以在获取到的原始问答数据中，包含着大量 HTML 标签和代码. 这些都会对语义分析产生影响. 同时，这些开源软件需遵守相关的开源协议.

对语句中出现的代码进行替换. 在图 2 所展示的问答记录中，存在灰色背景的命令行代码. 从网页获取的这段文本中，代码被包含在<code></code>标签，通过正则表达式<code>.*</code>将代码替换为 #CODE 标记.

对语句中出现的 HTML 标签进行替换，获取到的问答记录被包含在 HTML 标签之间. 用 Beautiful Soup 工具对出现的 HTML 标签进行替换.

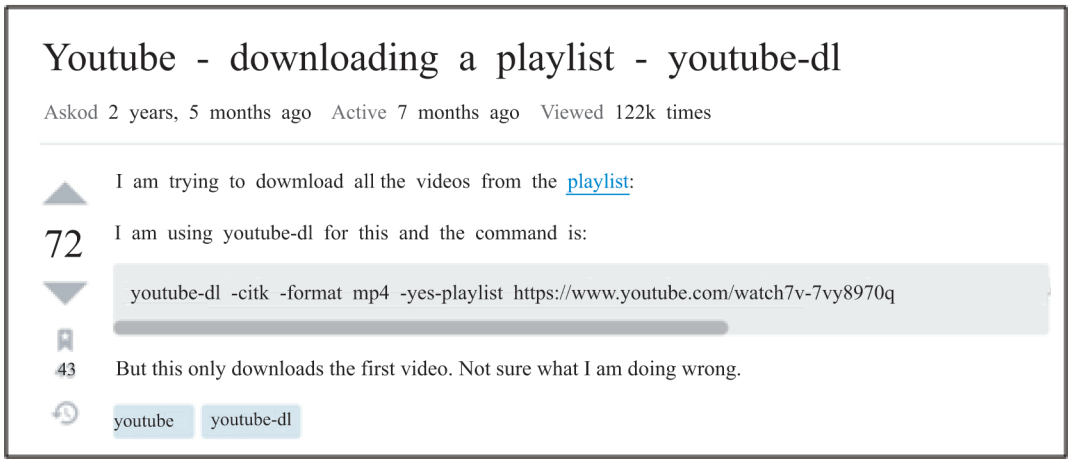

图 2 Youtube-dl 项目在 Stack Overflow 中的问答记录

Figure 2 Q&A record of youtube-dl in Stack Overflow

2.2 开源软件功能特征的筛选

根据文献 [15] 中提出的基于 Stack Overflow 数据的软件功能特征挖掘组织方法进行开源软件功能特征提取.

2.2.1 句法分析与动宾短语提取

在问答社区中，功能特征通常作为动宾短语出现. 因此用 Stanford CoreNLP[16]工具对语句构造句法分析树，并用 Penn Treebank[17]的约定进行语法与词性标记. 如语句“I'm trying to download all the videos from the playlist.”（来自开源项目 youtube-dl 在 Stack Overflow 中的讨论）的句法解析书构造如图 3 所示.

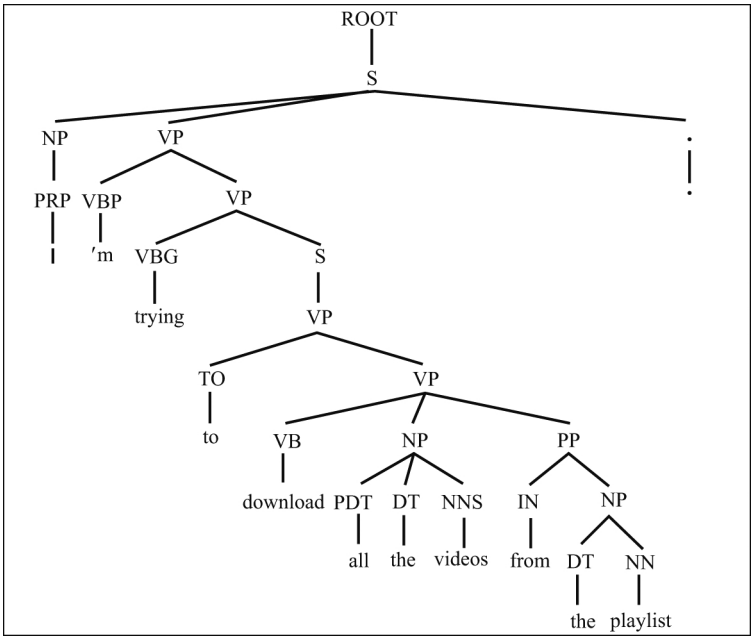

图 3　"I'm trying to download all the videos from the playlist."的句法解析树

Figure 3　Parse tree of the sentence of "I'm trying to download all the videos from the playlist."

在句法解析树中，VP 所对应的句子为动宾短语. 而文档中存在多个 VP 标记，只有"download all the videos from the playlist."为合法的功能特征. 因此设计了基于句法结构特征的过滤方式：如果动宾短语的第一层子节点不包含动词（标签 VB.*）和名词成分（标签.*NP.*）则直接过滤. 根据此过滤方法可以得到该句中合法的动宾短语作为候选功能特征.

2.2.2　基于启发式的动宾短语过滤

初步筛选出的动宾短语很多不能直接作为功能特征，需要通过启发式规则进行筛选. 本文在动宾短语筛选的过程中提出了核心词概念. 根据文献 [15] 所提出的功能特征的统一文法，如表 1 所示.

Function、Action、Object、Condition 分别表示功能特征、功能特征中的操作、功能特征中的对象、功能特征中的约束条件. 操作中的动词与对象最后一个名词分别对应着功能特征中的核心动词与核心名词. 如"download all the videos from the playlist"中，核心动词为 download，核心名词为 video，两者组合的核心词组为"download video".

表 1　功能特征的统一文法

Table 1　Normalized grammar of functional features

Function	::=	Action Object {Condition}
Action	::=	verb [particle]
Object	::=	{dt} {adj} {noun} noun
Condition	::=	prep [verb] Object

当核心动词出现助动词、情态动词，或软件开发常用术语（如 include 等）、问答常用术

语（如 ask，help 等）时，将短语过滤.

当核心名词出现代词或软件开发常用术语（如 method、object 等）、问答常用术语（如 solution、advice 等）时，将短语过滤.

2.2.3 基于核心词组出现频度的过滤

在问答讨论中会反复提及功能特征，因此通过过滤基于核心词组出现词频，能够有效筛选出合法的功能特征. 与文献 [15] 不同的是，本文方法统计动宾短语的核心词组在一个项目中出现的频度，当核心词组出现频度大于 0.01%，可被视为合法的功能特征. 由于本文方法无需获取开源软件功能的层次化组织结构，相较于文献 [15] 中使用的频繁子图挖掘算法，基于核心词组出现频度的过滤方式不仅能反映功能特征出现的频率，在处理大量数据时还能提高运行效率.

2.3 用户描述中功能特征的提取

针对用户上传的软件需求，通过 2.2 节中 1）、2）方法进行功能特征的提取. 比如，需求语句“I want to make a webpage，users can download youtube videos from the webpage.” 通过 2.2 节中 1）、2）方法，筛选出候选功能特征 “make a webpage” “download youtube videos using the webpage”.

由于用户需求描述文本中功能特征出现的频度较小，无法采用基于核心词出现频度的过滤. 因此为用户提供了一个允许其对提取、筛选后的候选功能特征进行删除、修改的接口，保证了软件需求中提取的功能特征的准确度，也避免了用户逐一查找的繁琐.

上文的候选功能特征“download youtube videos using the webpage”中，“from the webpage”对检索没有意义，通过修改接口修改为“download youtube videos”.

2.4 具有相似功能开源软件的检索与推荐

本文基于软件需求中的功能特征与开源软件中的功能特征，设计了如图 4 的检索与推荐流程.

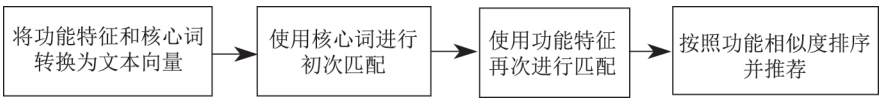

图 4 开源软件检索与推荐流程

Figure 4 Process of retrieval and recommendation

1）功能特征和核心词短语的文本向量转换

由于开源软件存在大量的功能特征且涉及领域不确定，无法提前获得数据集进行训练. 因此使用 Google 提供的 BERT-Large、Uncased (Whole Word Masking)预训练模型进行无监督的文本向量转换.

例如开源软件 youtube-dl 中具有的功能特征“download youtube videos”，其核心词组为“download video”，使用BERT将功能特征与核心词组分别转换为 1∗1 024 的文本向量.

2）使用核心词组进行初次匹配

由于开源软件中存在大量的功能特征，且不同的功能特征有可能存在相同的核心词. 因此先采用核心词组进行相似度匹配，能够有效减少检索的次数. 使用用户需求中提取出的核

心词组，对数据库中开源软件功能特征的核心词组进行相似度匹配，文本向量相似度检测如式(1)所示. 根据相似度进行排序筛选出前 J 个核心词组.

如用户所需功能为“download videos from youtube”，系统将先使用核心词组“download video”与数据库中的开源软件功能特征的核心词组进行相似度匹配，当匹配到 youtube-dl 项目中的核心词组“download video”，再与其对应的“download youtube videos”进行相似度计算.

3）使用功能特征进行再次匹配.

根据 2）中筛选出的核心词组所对应的功能特征，用式(1)检测短文本向量相似度. 根据相似度筛选出前 K 个开源软件的功能特征. 此处分为两个阶段检索是出于检索效率的考虑，通过核心词能够检索出具有较高推荐排名候选项目，即缩小二次检索范围，提高检索效率. 效率对比将在实验部分给出.

4）按照功能特征相似度进行排序，将排名前 K 个的软件项目推荐给用户.

3　实验评估

实验平台为 Intel i9 9880H 处理器，macOS 10.15.5 操作系统，运行内存 16 GB.

3.1　数据准备

我们筛选了 Github上Stars 数排名前 100 的 Python 开源软件. 从 Stack Overflow 上获取到相关的问答记录数据如表 2 所示. 有些开源软件在 Stack Overflow 社区中问答数目较多，例如 Flask 拥有 81 972 条问答记录. 由于实验设备性能限制，对于大于 10 000 条评论的开源项目，只截取其中浏览量最高的 5 000 条问题及其答案.

表 2　实验数据信息

Table 2　Information about experimental data

数据名称	数目	示例
开源软件	100	youtube-dl,flask,Django
开源软件在Stack Overflow的标签	128	[flask][scrapy][web-scrapy]
问答语句	332 697	How to solve link Posts and users to comments in Django Admin

3.2　评价指标与实验设计

首先，采用 MRR（mean reciprocal rank）[18]对项目推荐性能进行评估. MRR是检索系统中常用的评估指标. 检索系统根据条件生成一个项目排名列表，MRR 是第一个符合条件的项目在列表中排名的倒数乘积. MRR 的值越大，说明该系统的性能越好.

由于开源软件库中软件数目多，软件所具有的功能特征也较多，采用人工方法对所有软件的功能特征进行标定工程量较大，无法使用 Recall 等指标进行衡量，因此采用 MRR 来衡量系统的检索性能. 根据用于测试的软件项目编写用户需求描述，假设测试项目为开源软件F，其用户需求描述为 Q. 然后，用 Q 作为检索条件获得一个推荐项目列表. 由于 Q 可能不只是项目 F 独有的功能，检索结果中会有同样满足 Q 要求的非 F 项目. 为了消除该类项目对评价检索方案的影响，我们将其从列表中删除，仅保留并不满足 Q 要求的开源项目. 最后，根据 F 在处理后的推荐列表中的排名计算 MRR 值. 假设对于给定的查询 Q，其检索结果列表为

$D=\{\langle F_1,Q_1\rangle,\langle F_2,Q_2\rangle,\langle F_3,Q_3\rangle,\cdots,\langle F_{|D|},Q_{|D|}\rangle\}$，其 MRR 的计算为

$$\mathrm{MRR}=\frac{1}{|D|}\sum_{i=1}^{|D|}\frac{1}{\mathrm{Rank}_i} \tag{2}$$

式中，Rank_i 表示对于查询 Q_i，开源软件 F_i 的排序结果. MRR 的值越高，则项目推荐性能越好.

为了体现先用核心词组进行检索的效率优势. 在实验中，设置了分段检索与一次检索的平均检索耗时（单位：ms）的对比. 分段检索为先用核心词进行初次检索，再用核心词组所对应的完整的功能特征进行检索. 一次检索为直接使用需求中所需的功能特征与提取出的开源软件的功能特征进行文本向量相似度匹配.

为了评估本文提出的方案的有效性，设计了两个对比实验. 第一个实验是与基本检索方案对比检索性能（Baseline）. 基本方案为使用不包括本文构造的功能描述作为项目索引的检索方案，如基于项目简介、标签等信息进行检索. 最后用同样的方式计算测试项目的 MRR 值进行对比. 第二个对比实验是对比本文分阶段检索（Split）与一次检索（Single）的差异，包括推荐效果差异和效率差异.

表 3 用于测试的开源软件的用户需求样例

Table 3 Examples of user requirements of open source software for testing

测试软件名称	需求描述样例	
	功能 1	功能 2
youtube-dl	Users can download youtube videos.	Users can download audio from the playlist.
httpie	The tester will send a post request.	They will post a multipart/form-data request with a file.
scrapy	We need to scrape content from websites.	Change IP address dynamically when crawl websites.
spaCy	Remove custom stop words.	Extract verb phrases.
tornado	Use non-blocking network I/O in the application.	Build an asynchronous RESTful web service.
incubator-superset	Upload a CSV to your database.	Query and visualize your data with SQL Lab.
certbot	Install SSL certificate to my site.	Users can getet SSL certificate from cerbot.
pandas	Load data from a CSV file.	Sort by an axis.
django-rest-framework	Create a simple API to allow admin users to view.	I am trying to record and upload an audio to django server using rest framework.
fastapi	Offer a web interface to build Interactive API documentation.	Send data from a client to your API.

3.3　实验结果与分析

由于数据集中的开源项目均为面向开发者用户的 Python 平台项目，我们邀请了5 位熟练的 Python 开发工程师并随机选取了 10 个开源项目. 每个选取的开源软件由 3 位工程师独立编写人工描述的需求，并对 3 份不同的描述分别进行实验取平均值. 表 3 给出了每个项目的用户需求示例.

将 MRR 中的 K 设置为 20，针对每条需求中所描述的功能，每次从检索结果中抽取出 20 个不符合功能描述要求的开源软件，并统计该开源软件在这 $K+1$ 个软件中的排名. 针对每个用于测试的开源软件，共进行 3 次实验取平均值. 图 5 描述了不同开源软件在不同检索方案下对应的 MRR 评分.

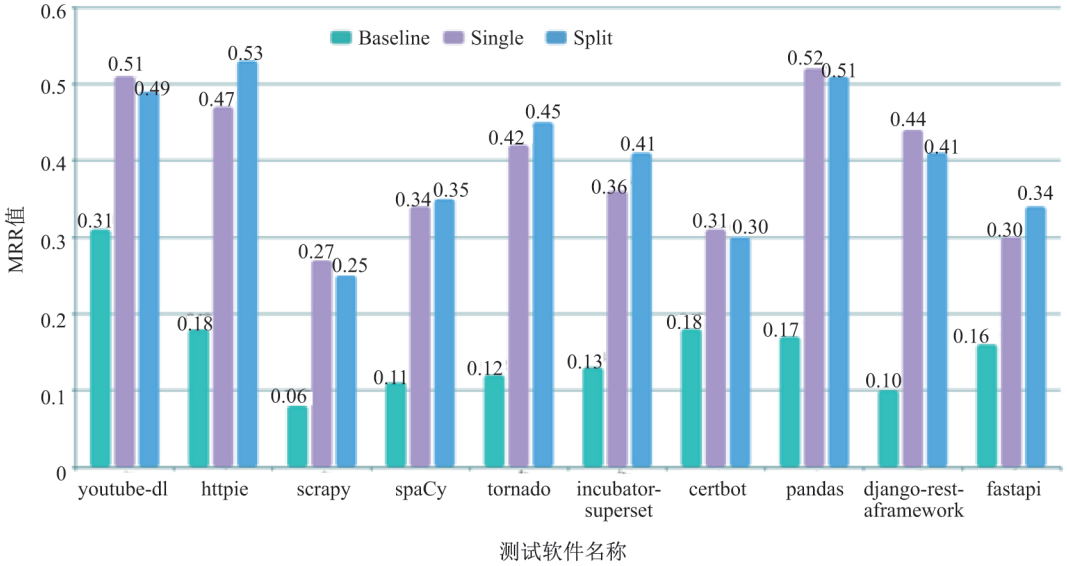

图 5　不同开源软件的不同检索方案对应的 MRR 评分

Figure 5　MRR of different open source software using different searching strategies

根据图 5 可以得出，Split 比 Baseline 的实验效果更好，但与 Single 没有明显差别. 在 Split 与 Baseline 的对比中，由于大部分开源软件（如 scrapy、spaCy）的描述、简介中没有提及软件功能，所以 Split 实验效果远优于 Baseline. 少部分开源软件（如 youtube-dl）的描述中涉及了大部分软件功能，所以其 Baseline 实验效果要远优于在其他开源软件上 Baseline 的实验效果. 在 Split 与 Single 对比中，由于两种方法最终都要进行功能特征的匹配，所以实验效果没有明显差别.

第二个实验中我们同样用上述 10 个测试软件项目，使用多个功能描述进行多次检索，统计平均检索效率，对比结果见表 4.

如表 4 所示，分阶段检索的平均检索耗时明显少于一次检索的平均检索耗时. 因为一个核心词组（如 download video），往往对应着多个功能特征（如download videos from youtube、download videos from playlist、download playlist videos等），通过核心词组可以快速定位到对应的功能特征，再进行第二阶段的检索. 因此在软件检索过程中，采用分阶段检索的方法所需的计算更少，所需时间更少.

表 4 分阶段检索和一次检索的效率对比

Table 4 Comparison of searching efficiency between splite and single strategy ms

测试软件名称	平均检索耗时	
	Split（分阶段）	Single（一次检索）
youtube-dl	1 954	9 516
httpie	1 308	6 544
scrapy	984	4 816
spaCy	1 104	5 861
tornado	1 298	6 652
incubator-superset	426	2 045
certbot	511	2 038
pandas	1 427	7 164
django-rest-framework	1 208	6 863
fastapi	753	3 526

3.4 系统实现

基于本文提出的方法，我们实现了一个供用户基于软件功能需求描述的相似软件检索系统. 通过使用 Flask 框架与 Vue.js 框架进行前后端的搭建. 如图 6 所示，该系统为用户提供了一个用户需求文档上传的界面. 用户可以将需求分析以 txt 文档格式上传. 系统将自动抽取出用户所需的候选功能并支持用户进行修改、删除.

图 6 软件需求上传编辑界面

Figure 6 Interface of uploading software requirements and editing functions

如图 7 所示，搜索结果将以列表的形式呈现给用户. 根据软件需求中提取、筛选出的功能为用户推荐具有相似功能的开源软件，并给出相应的链接与开源证书.

4 结 语

本文提出了基于软件需求的开源软件检索方法，从开源软件在问答社区的问答记录中提取开源软件的功能特征，能够有效解决开源软件社区中软件标签较少、软件描述不全对开源

软件检索造成的负面影响. 通过文本向量对功能进行检索，在一定程度上解决了功能描述上的差异问题. 同时，系统对软件需求中的功能需求进行批量抽取与检索，避免了开发者在开发过程中逐个检索的繁琐. 因此，本方法在开源软件检索上有一定的价值.

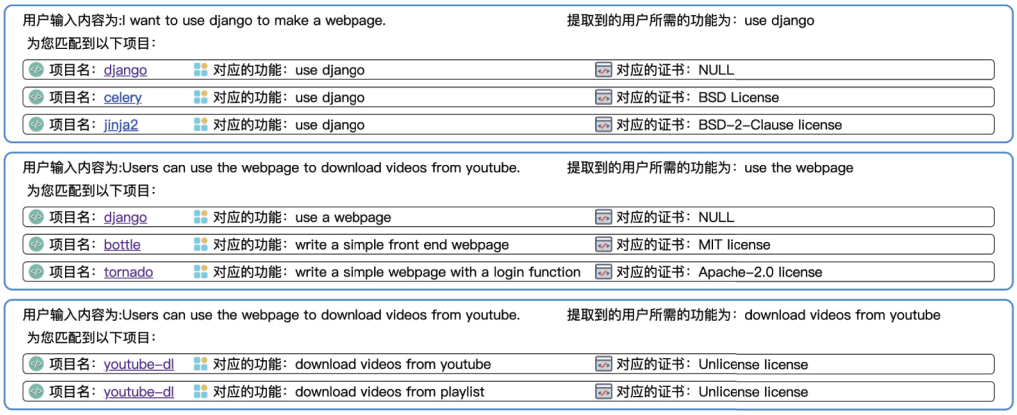

图 7　系统查询结果界面

Figure 7　Query result interface of the system

由于很多开源软件在问答社区中问答数目较少，会出现部分功能不能被体现的情况，很大程度上影响了检索的效果. 后续实验中将会从代码信息，注释信息等更多地方获取开源软件信息，作为检索的条件.

参考文献:

[1] Bajracharya S, Ngo T, Linstead E, et al. Sourcerer: a search engine for open source code supporting structure-based search [C]//Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications, 2006: 681-682.

[2] Gu C, Yin G, Wang T, et al. A supervised approach for tag hierarchy construction in open source communities [C]//Proceedings of the 7th Asia-Pacific Symposium on Internetware, 2015: 148-152.

[3] Girardi M R, Ibrahim B. Using English to retrieve software [J]. Journal of Systems and Software, 1995, 30(3): 249-270.

[4] Paul S, Prakash A. A framework for source code search using program patterns [J]. IEEE Transactions on Software Engineering, 1994, 20(6): 463-475.

[5] Zhang Y, Lo D, Kochhar P S, et al. Detecting similar repositories on GitHub [C]//2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2017: 13-23.

[6] Devlin J, Chang M W, Lee K, et al. Bert: pre-training of deep bidirectional transformers for language understanding [DB/OL]. 2018[2019-05-24]. https://arxiv.org/abs/1810.04805.

[7] Frakes W B, Kang K. Software reuse research: status and future [J]. IEEE Transactions on Software Engineering, 2005, 31(7): 529-536.

[8] Haefliger S, Von Krogh G, Spaeth S. Code reuse in open source software [J]. Management Science, 2008, 54(1): 180-193.

[9] Cossentino M, Burrafato P, Lombardo S, et al. Introducing pattern reuse in the design of multi-agent systems [C]//Net.ObjectDays: International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World. Berlin: Springer, 2002: 107-120.

[10] Palomares C, Franch X, Quer C. Requirements reuse and patterns: a survey [C]//International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, Cham, 2014: 301-308.

[11] Arango G F. Domain engineering for software reuse [M]. Irvine: University of California, 1988.

[12] Barreto A, Murta L G P, da Rocha A R C. Software process definition: a reuse-based approach [J]. Journal of Universal Computer Science, 2011, 17(13): 1765-1799.

[13] Stewart K J, Ammeter A P, Maruping L M. Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects [J]. Information Systems Research, 2006, 17(2): 126-144.

[14] Lerner J, Tirole J. The scope of open source licensing [J]. Journal of Law, Economics, and Organization, 2005, 21(1): 20-56.

[15] 朱子骁, 邹艳珍, 华晨彦, 等. 基于 StackOverflow 数据的软件功能特征挖掘组织方法 [J]. 软件学报, 2018, 29(8): 2210-2225.
Zhu Z X, Zou Y Z, Hua C Y, et al. Mining and organizing software functional features based on StackOverflow data [J]. Journal of Software, 2018, 29(8): 2210-2225. (in Chinese)

[16] Manning C D, Surdeanu M, Bauer J, et al. The stanford CoreNLP natural language processing toolkit [C]//Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014: 55-60.

[17] Marcus M, Santorini B, Marcinkiewicz M A. Building a large annotated corpus of English: The Penn Treebank [J]. Computational Linguistics, 1993, 19(2): 313-330.

[18] Voorhees E M. The TREC-8 question answering track report [C]// Text Retrieval Conference, 1999, 99: 77-82.

(编辑：王 雪)