

代码搜索与 API 推荐文献分析

聂黎明 江 贺 高国军 王 涵 徐秀娟

(大连理工大学软件学院 大连 116621)

摘 要 代码搜索和 API 推荐算法能够帮助开发者有效实现编程任务。截至目前,研究者们发表了一系列相关文献。尽管一些学者对该研究领域的背景和研究现状进行了阐述,但是研究者对该领域中的一些基本领域知识还缺乏了解,如最高产的作者、机构和国家,影响力较大的作者和文献,以及流行的热点研究等。借助经典的文献分析框架,在构建该研究领域文献数据仓库的基础上,首次对该领域的研究进行了基础文献分析和合作模式探索。一方面,基础文献分析的结果表明,近几年越来越多的研究者开始关注该领域的研究,最高产的作者是 Cristina Videira Lopes, University of California at Irvine 是发表相关文献最多的机构,大部分文献来自美国,根据领域 H 因子计算得到的最有影响力的作者是 Denys Poshyvanyk。另一方面,合作模式的分析结果显示, Tao Xie, Cristina Videira Lopes 和 Denys Poshyvany 是该领域最活跃的三位作者,推荐算法性能的提升及其在软件工程任务中的应用是目前该领域最流行的研究主题。

关键词 文献分析,代码搜索,API 推荐,合作,研究热点

中图法分类号 TP311.5 **文献标识码** A

Bibliographic Analysis for Code/API Recommendation Literatures

NIE Li-ming JIANG He GAO Guo-jun WANG Han XU Xiu-juan

(School of Software Technology, Dalian University of Technology, Dalian 116621, China)

Abstract Code/API recommendation approaches can assist developers to implement programming tasks efficiently. Until now, lots of related literature are published. Although some researchers present the background and the state of this research field, there is a lack of understanding of some essential domain knowledge, such as the most productive author, institution, and country, the most popular literature and author, and the popular research hotspots. By employing a classical bibliographic analysis framework, we conducted a basic bibliographic analysis and explored cooperation among authors based on literature data. The analysis results show that, on the one hand, in the basic bibliographic analysis, we found that the most productive author is Cristina Videira Lopes. The University of California at Irvine is the most productive institution, most of literature are from the USA, and the most influential author is Denys Poshyvanyk. On the other hand, in the research about cooperation among authors, we found that three authors, i. e., Tao Xie, Cristina Videira Lopes, and Denys Poshyvany are the most active authors in this research field. Two topics, i. e., the performance improvement for recommendation algorithms and their applications on other software engineering tasks, are the most popular research topics.

Keywords Bibliographic analysis, Code search, API recommendation, Cooperation, Research interests

1 引言

代码搜索技术能够辅助开发者较快完成手头上要实现的编程任务。对于某个代码搜索工具,开发者要实现某个特定的编程任务时,如果他了解实现该任务的相关 Application Program Interfaces(APIs),但不清楚 APIs 的具体使用情况,则可以在搜索工具中输入 API 的名字,搜索工具能够从事先准备好的代码段仓库中推荐相关的代码片段。开发者如果事先并不知道要使用哪个(或哪些)API,也可以输入自由文本

的查询、当前编写程序的上下文或者测试用例等,搜索算法也能够推荐实现该功能的相关代码片段或相关的 APIs。为了实现更高效、快捷、准确的推荐,最近几年出现了一系列相关文献。研究者在阅读这些相关文献时,可能想要了解一些基本的领域知识,比如,哪篇文章是最有影响力的文章?高产的作者、机构和国家是哪些?该领域最有影响力的作者是哪位?作者之间是如何合作的?该领域流行的研究主题等。截至目前,在该研究领域已经发表了一些类似综述的文献。但是,这些文献要么偏重于介绍搜索算法的各个环节^[1],要么只是少

本文受国家自然科学基金项目(61370144),教育部新世纪优秀人才计划(NCET-13-0073)资助。

聂黎明(1983—),男,博士生,主要研究方向为代码搜索, E-mail: limingnie@live.cn; 江 贺(1980—),男,博士,教授,CCF 会员,主要研究方向为软件仓库挖掘、基于搜索的软件工程, E-mail: jianghe@dlut.edu.cn(通信作者); 高国军(1992—),男,博士生,主要研究方向为软件仓库挖掘; 王 涵(1995—),女,硕士生,主要研究方向为软件仓库挖掘; 徐秀娟(1980—)女,博士,副教授,CCF 会员,主要研究方向为大数据分析、智能交通系统。

数几个 APIs 推荐算法的对比^[2], 缺乏对该研究领域整体概况的分析, 无法较好地回答上述问题。

为了解决这一问题, 借助于经典的文献分析框架^[3]为代码搜索和 API 推荐(以下简称为: code/API 推荐)领域构建了一个文献分析研究。该研究包括 3 个方面: 首先, 从 IEEE Xplore, EI Compendex, ISI Web of Science (WoS), ACM Digital Library, Google scholar, DBLP^[1] 等数据仓库中收集与该领域研究相关的文献数据, 数据收集截止到 2016 年 8 月。其次, 基于这些文献数据, 我们进行了一个基本的文献分析来探究历年的论文发表情况, 发现最高产的作者、机构和国家, 最有影响力的文献和作者, 以及常见关键词频率等。最后, 通过构建和分析 3 个网络来探究该研究领域的合作模式、研究热点和作者共同的研究兴趣。

在基础文献分析中发现, 该领域研究最早的一篇论文发表于 1987 年, 然而, 从 2005 年开始相关的文献才逐渐增多。2011 年—2014 年的相关论文发表得相对较多。可以看出, 近几年越来越多的研究者开始关注该领域的研究。另外, 经过统计分析发现, 作者 Cristina Videira Lopes^[4] 是发表论文最多的作者, 她在该研究领域共发表了 15 篇文章; University of California at Irvine 是发表文献最多的机构, 该机构共发表了 21 篇文章; 该领域的相关文献大部分来自美国的研究机构。经 NCII 指数计算得到, 该领域最有影响力的文献是发表于 2007 年的“Parseweb: a programmer assistant for reusing open source code on the web”^[5], 截至目前, 该论文被引 311 次; 用领域 H 因子计算得到该领域最有影响力的作者是 Denys Poshyvanyk。

另外, 为了探究作者之间的合作模式等信息, 我们构建了 3 个网络, 即作者共著网络 (Co-authorship Network)、关键词共现网络 (Keyword Co-occurrence Network) 以及作者共关键词网络 (Author Co-keywords Network)。通过使用 GN (Girvan-Newman) 算法^[6] 对这 3 个网络中隐藏的社区进行分析, 分别呈现该研究领域涉及到的研究者之间的合作关系、该领域流行的研究主题以及作者之间共同的研究兴趣。分析结果显示, Tao Xie^[5], Cristina Videira Lopes^[7] 和 Denys Poshyvanyk^[8] 3 位作者是该领域最活跃的作者, 该领域的研究主要围绕提升推荐算法的性能及其在软件工程具体任务中的应用展开。

本文的主要贡献如下:

- 1) 共收集了 218 篇 code/API 推荐相关的文献, 从每篇文献的链接网站上收集了相关的文献数据。
- 2) 对该领域构建了一个基础的文献分析研究来发现该领域历年的论文发表情况, 最高产的作者、机构和国家, 最有影响力的文献和作者, 常见关键词的频率等。
- 3) 根据收集到的文献数据, 构建了 3 个网络。利用经典的网络分析算法来探究作者之间的合作关系、相关文献中流行的研究主题以及作者间共同的研究兴趣。

本文第 2 节介绍相关工作; 第 3 节展示文献分析的总体过程; 第 4 节阐释基础文献分析的结果; 第 5 节呈现 3 个网络的构建及结果分析; 最后总结全文。

2 相关工作

本部分将介绍一些相关的工作, 包括代码搜索和

API 推荐以及文献分析研究。

2.1 代码搜索和 API 推荐

按照代码搜索方法的输入类型划分, 相关文献可分为以自由文本作为输入的研究、以 API 作为输入的研究、其它形式作为输入的研究。而推荐的内容要么是相关的代码片段, 要么是单个 API, 或 APIs 的调用序列。这里, 把推荐代码的研究称为代码搜索, 把推荐 API 及其序列的研究称为 API 推荐。

在代码搜索研究中, 以自由文本作为查询, Zhong 等人^[9] 提出为开发者的查询推荐 API 使用模式及对应的代码片段。Bajracharya 等人^[4] 在 Lucene 的多域搜索平台上, 综合 API 使用模式的相似性等一系列特征进行代码片段的推荐。McMillan 等人^[10] 在信息检索的基础上, 利用 PageRank 和传播模型推荐具有依赖关系的函数。Keivanloo 等人^[11] 利用向量空间模型、频繁项挖掘等技术为自由文本的查询进行代码推荐。Raghothaman 等人^[12] 提出了一种新的方法 SWIM, 该方法首先从用户点击数据中找到与用户输入的自由文本查询相关的 APIs, 同时从 GitHub 的代码中抽取出结构化的 API 调用序列, 通过匹配找到 APIs 相应的调用序列, 进而产生合成的代码片段。以上提到的方法都是基于信息检索的技术。不同于这些方法, Jiang 等人于 2016 年尝试使用监督学习的方法为自由文本的查询提供相关的代码片段^[13]。另外, Li 等人^[14] 提出了基于关系的代码搜索方法, 在该方法中, 代码片段中的 API 使用模式和用户的查询分别被表示为方法之间的关系调用图和行为关系图。这样, 代码搜索问题就被转换为了图搜索问题。

在以信息检索为主要匹配方法的代码搜索研究中, 常常因为查询词过短以及查询和匹配的代码片段使用不同的语言而导致词项失配的问题。为了解决这一问题, 研究者提出了一系列的基于查询扩展的代码搜索方法。例如, Wang 等人^[15] 利用半自动的相关反馈方法为代码搜索引擎推荐的代码段列表进行评价, 而后再根据评价结果进行重排, 进而得到了更好的推荐结果。另外, 研究者也考虑从 WordNet 中抽取出近义词来扩展查询^[16] 或测试用例^[17]。为了获取与软件开发和编程相关的近义词, 研究者尝试从 Stack Overflow 的编程问答对中抽取出相关的近义词^[18] 或 API^[19] 来扩展原始查询, 并取得了好的效果。

在以 API 作为查询的代码搜索研究中, Subramanian 等人^[20] 提出了 Baker 方法, 其利用代码片段来丰富 API 文档; Moreno 等人^[21] 提出了 MUSE 方法, 该方法利用静态切片和代码克隆技术, 从一系列相似的代码片段中总结出一套调用某个 API 的步骤并呈现给开发者。

在以其他类型内容作为查询的代码搜索研究中, Lemos 等人^[22] 提出以测试用例作为输入进行搜索, 其目的是把开发者的搜索需求尽可能地在测试用例中描述清楚; Stolee 等人^[23] 以开发者期待的某段代码的输入和输出状态的配对作为输入来推荐代码片段; Inoue 等人^[24] 提出 IchiTracker, 其以开发者当前编写的代码片段作为输入, 推荐与输入相符的开发者可能需要的其他代码。

除了推荐代码片段, 也有较多学者关注 APIs 的推荐。例如, Thung 等人^[25] 提出从版本控制系统的代码历史变更信

^[1] <http://dblp.uni-trier.de/db>

息中抽取知识,从而推荐相关的 API,Gu 等人^[26]提出了一种基于深度学习的 API 序列推荐方法——DeepAPI。该方法把 API 推荐问题转换成监督学习问题,把用户输入的自由文本查询转换成相关的 API 序列。类似地,Niu 等人^[27]提出了使用监督学习的方法,在两阶段的基础上,借助于多种特征进行 API 的推荐。另外,Rahman 等人^[28]尝试从 Stack Overflow 的问答对中抽取与开发者的自由文本的查询相关的 API。这些工作都取得了较好的效果。

上述文献要么通过新技术来提升算法的性能,要么通过使用一些领域信息辅助代码或 API 的推荐。本文研究的主要目的是通过对该领域相关文献的分析,为研究者呈现该领域的一些基本的领域知识以及研究者之间的合作情况等。

在本文中,将 code/API 推荐的相关文献一起进行分析,是因为表面上这两方面的研究的输出类型截然不同。代码搜索算法致力于把代码片段推荐给开发者,以方便开发者通过复制、粘贴和修改的方式实现代码重用。而 API 推荐算法致力于为开发者推荐单个或多个 API,以及调用序列。实际上,两个方面研究的目标是一致的,都是为了辅助开发者的开发,便于其能快速实现相应的编程任务。另外,一些研究在推荐结果中不仅提供 API,还提供使用这些 API 的代码片段^[9]。因此,统一分析两个方面的研究是有必要的。

2.2 文献分析

文献分析研究主要采用数理统计和数据挖掘等方法对某个特定领域的文献进行深度挖掘,把该领域的研究背景、历史和现状以及流行的研究主题等领域信息呈现给该领域的研究者^[29-30]。近几年,研究者做了大量针对软件工程领域文献的分析^[31],例如,Wohlin 做了一系列的工作来分析软件工程期刊上引用率最高的文献^[32]。Cai 等人通过调研几个重要软件工程期刊和会议上的研究主题来分析研究者的研究重点^[33]。Fernandes 对 1971 年—2012 年的 7 万多篇文献中涉及的作者进行了分析,以探究作者对文献的贡献^[34]。Bornmann 使用文献的被引次数和年平均被引次数来发现软件工程研究中高被引文献^[35]。另外,Xu 等人对智能交通领域的研究进行了一个较为全面的文献分析研究^[3]。本文首次构建了对 code/API 推荐文献分析的研究。不同于以往仅从单一角度出发对软件工程领域进行文献分析,本文不仅分析了高产的作者、机构和国家,还分析了高影响力的作者和文献等,同时还构建和分析了 3 个网络来探究作者之间的合作模式、共同的研究兴趣等。

3 总体研究过程

本节将介绍本文研究的 3 个主要方面:1)收集该研究领域相关的文献数据;2)构建基本的文献分析;3)分析作者之间的合作关系。

3.1 收集文献数据

通过以下 3 个阶段收集该研究领域的文献数据^[36]:识别领域关键词、过滤相关文献和抽取文献数据。

首先,为了在网络搜索 code/API 推荐的相关文献,从现有的 86 篇文献中识别了 22 个领域关键词及其近义词。这

些识别的词包含代码搜索和 API 推荐两类,例如:code search,API recommendation。在下一阶段,这些词将被布尔或类型连接后形成查询输入,例如,上述两个关键词的组合是 code search OR API recommendation。

其次,如图 1 所示,使用两步来过滤相关文献。在第一步中,对 4 个网上数据仓库进行手动搜索。这 4 个数据仓库分别是:IEEE Xplore, EI Compendex, ISI Web of Science (WoS), ACM Digital Library。搜索时使用的输入是上个阶段合成的字符串,搜索的范围是标题、摘要和关键词。通过这一步骤,共收集了 1082 篇候选文献。根据两条过滤规则,以达到人工消除重复和不相关文献为目的,最终共得到 141 篇相关的文献。这两条过滤规则分别是:文献中不包括软件工程或软件开发相关的词,以及文献中不包括代码搜索或 API 推荐相关的词。在第二步中,根据得到的相关文献进一步获取关联的其他相关文献。第二步的主要目的是保证不会忽略掉任何相关的文献,主要的方法是先扫描得到的相关文章的参考文献列表,然后根据上述提到的两条过滤规则来识别相关文献,最终得到 77 篇新的相关文献。根据以上两个步骤,截止到 2016 年 10 月 1 号,总共收集了 218 篇文献。再次手工确认这些文献,确保这些文献都是与 code/API 推荐研究相关的。

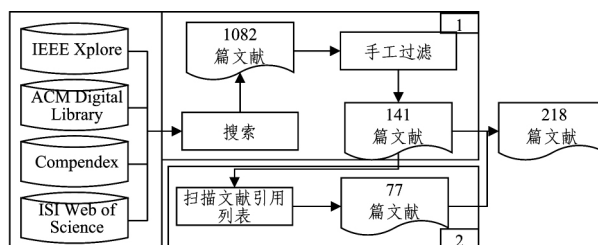


图 1 两个阶段的搜索步骤及对应的文献数量

最后,在确定了相关文献的标题后,将抽取某个文献标题对应的文献数据。具体来说,给定一个文献的标题,要抽取的信息包括:文章的关键词、引用数、作者的名字、机构和国家等。为了收集这些数据,首先在 crossref¹⁾网站中输入某篇文章的标题,可以反馈得到这篇文章的 DOI 网页链接;根据得到的链接,使用一个网络爬虫获取相应的文献数据。鉴于有些作者的名字在 IEEE Xplore 网站上使用的是缩写,从 DBLP 网站(一个知名的计算机研究相关的网站)数据库中获取这些作者的全名。此外,从 Google Scholar 网站上获取每一篇文章的引用次数。获取文献数据的日期截止到 2016 年 8 月 1 日。下面要进行的基础文献分析和合作模式探索都是基于这些收集到的文献数据。本文在自己的网站²⁾上提供了收集到的文献数据的下载链接。

3.2 基础文献分析的准备

基于收集的文献数据,采用文献^[3]中的经典文献分析方法对 code/API 推荐研究领域进行了基础的文献分析。分析内容包括:该领域历年文献发表数量的变化情况,最高产的作者、机构和国家,最有影响力的文献和作者,以及关键词统计分析。

¹⁾ <http://search.crossref.org/>

²⁾ <http://oscar-lab.org/people/~lnie/BA/>

其中,在进行文献的影响力分析时,引入指标 NCII(Normalized Citation Impact Index)。一般情况下,论文的引用数量与其发表时间有较大关系。一篇文章发表的时间越早,其被引用的次数可能就越多。在对比文献的影响力时,直接对比两篇文献的引用次数显然是不合理的。考虑到出版时间对引用次数的影响,Serenko 等人^[37]提出了 NCII 指标。计算某篇文献的 NCII 值的公式如下:

$$NCII = \frac{\text{the literature's citation}}{\text{literature longevity(in year)}} \quad (1)$$

其中,“the literature's citation”指该文献的引用次数,“literature longevity(in year)”指文献发表的时间长度。可以看出,实际上 NCII 指数计算的是文献每年的平均引用次数,比直接使用引用次数来度量文献的影响力显得更加客观。

另外,在评价作者影响力时,引入 H 因子(H Factor)^[38],又称 H 指数,它被广泛用于度量不同领域作者的影响力。对于某个作者,如果他发表的论文越多,同时论文的被引用次数越多,那么他的 H 因子也就越高。很多作者的 H 因子都可以在 Google Scholar 中查询到。但是,本文不直接使用 Google Scholar 中的 H 因子,因为其衡量的是某作者在所有研究领域的影响力,本文需要计算的是作者在 code/API 推荐领域的 H 因子,然后根据 H 因子的大小进行排序,具体做法是,对于某位作者,将他发表的该领域的文献根据被引用次数进行排序,排序后每篇文献对应一个序号,将该序号与该文献的被引用次数进行比较,找到某个序号 h 的论文,满足该序号的值小于或等于该篇论文的被引次数,而序号 $h+1$ 的文献大于它的被引次数。这样,序号 h 就是该作者的领域 H 因子值。

3.3 合作关系探索的准备

首先,为了探究作者之间的合作关系、相关文献的研究主题以及作者之间共同的研究兴趣,构建了 3 个网络作者共著网络(Co-authorship Network)、关键词共现网络(Keyword Co-occurrence Network)和作者共关键词网络(Author Co-keywords Network);然后,使用一个经典的 GN(Girvan-Newman)算法^[39]来分析这 3 个网络中隐藏的社区结构;最后,使用软件 UCINET^[40]对社区结构进行可视化呈现。

表 1 列举了 3 个网络的一些细节信息,其中,AN 指作者共著网络,KN 指关键词共现网络,AKN 指作者共关键词网络,3 个网络均为无向图。可以看到,作者共著网络和作者共关键词网络中的节点表示的是作者,两个网络中的边分别表示两个作者共同发表过一篇文章或者共同使用过一个关键词。而关键词共现网络中的节点表示的是关键词,两个节点的连线表示的是两个关键词出现在一篇文章中。

表 1 3 个网络的节点和边的表示

网络	节点	边
AN	作者	两个作者共同发表过一篇文章
KN	关键字	两个关键词出现在一篇文章中
AKN	作者	两个作者使用过同一个关键词

4 基础文献分析

本节将介绍基于收集到的文献数据进行的基础文献分析及结果。如前所述,本文共收集了 218 篇与 code/API 推荐研

究相关的文献。这些文献共包含了 415 个作者,这些作者分别来自 27 个国家或地区的 163 个机构。下面分别从生产力、影响力和关键词 3 个方面进行分析。

4.1 历年文献数量分析

为了探究文献发表数量随时间变化的情况,统计了每年发表的文献数目。图 2 示出了自 1987 年开始到 2016 年 10 月该研究领域每年的文献发表数目。可以看到,该领域第一篇研究工作发表于 1987 年,之后几年只有零星文献发表,直到 2005 年,相关文献的数目才开始逐渐增加;同时,2011 年到 2014 年的这 4 年的文献发表数明显比其他年份明显多,其中 2011 年达到最大,共有 31 篇文章发表。这表明,近几年越来越多的学者开始关注该研究领域。

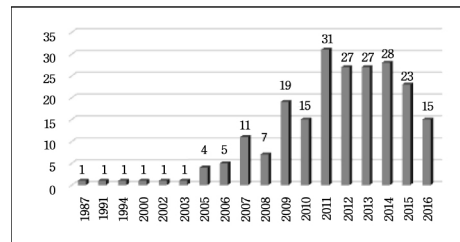


图 2 历年文献发表的数量

4.2 产量分析

本部分内容包括最高产的作者、机构和国家的分析。首先,在统计每个作者的相关文献时,同一个作者的名字在多个文献中可能不一致,统一使用该作者的全名,例如,对于同一作者的两个名字 Eduardo S. de Almeida 和 Eduardo Santana de Almeida,统一用 Eduardo Santana de Almeida 来表示。表 2 列出了前 10 位高产的作者。其中,排在第一的是 Cristina Videira Lopes,她发表过 15 篇相关文献。排在第二和第三位的分别是 Sushil Krishna Bajracharya 和 Denys Poshyvanyk,他们分别发表了 12 篇和 10 篇相关文献。

表 2 前 10 名高产的作者

排名	作者	文献数量
1	Cristina Videira Lopes	15
2	Sushil Krishna Bajracharya	12
3	Denys Poshyvanyk	10
4	Collin McMillan	9
5	David Lo	8
6	Mark Grechanik	8
7	Tao Xie	7
8	Joel Ossher	7
9	Otavio Augusto Lazzarini Lemos	7
10	Kathryn T. Stolee	6

其次,为了探究最高产的机构,统计了每个机构发表文献的数量。具体的方法是,如果同一篇文章包含的多个作者来自不同的机构,那么这些涉及到的机构的文献发表数量都加 1;如果作者中有两个或多个来自同一机构,则该机构的发表数量只加 1 次。表 3 展示了前 10 个最高产的机构。因为一些学校拥有多个校区,这些校区的学术水平有一定差别,所以对它们进行区别统计。从表 3 中可以看到,发表文章数量最多的机构是 University of California at Irvine,排在第二和第三的机构分别是 North Carolina State University 和 College of William and Mary。

表 3 前 10 个高产的机构及其所属国家

排名	机构	数量	国家
1	University of California at Irvine	21	美国
2	North Carolina State University	12	美国
3	College of William and Mary	11	美国
4	Iowa State University	9	美国
5	Singapore Management University	9	新加坡
6	Accenture Technology Labs	6	美国
7	Brown University	6	美国
8	Hong Kong University of Science and Technology	5	加拿大
9	ABB Corporate Research	5	美国
10	University of Delaware	4	美国

最后,与机构发表文献统计类似,我们能获取每个国家发表文献的情况。具体的方法是,如果同一篇文章包含的多个作者来自不同的国家,那么这些涉及到的国家的文献发表数量都加 1;如果作者中有两个或多个来自同一国家,则该国家的发表数量只加 1 次。图 3 展示了前 10 个高产的国家,从中可以看到,发表相关文献最多是美国,有 108 篇相关文献,其远远超过第二名的中国,中国只有 33 篇相关文献。可以看出,美国、中国、加拿大的研究者贡献了大部分的文献。

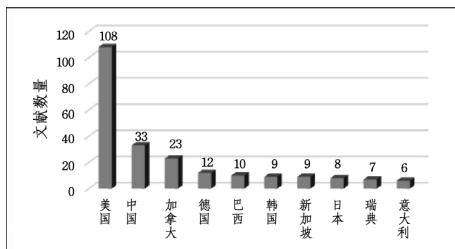


图 3 前 10 个文献发表数量较多的国家

4.3 影响力分析

4.3.1 文献影响力分析

表 4 列出了影响力较高(NCII 值较大)的前 10 篇文献。

表 4 前 10 篇高影响力文献

排名	论文	被引次数	NCII	发表年份
1	Parseweb: a programmer assistant for reusing open source code on the web	311	34.6	2007
2	Jungloid mining: helping to navigate the API jungle	353	32.1	2005
3	Example-centric programming: integrating web search into the development environment	174	29	2010
4	Using structural context to recommend source code examples	298	27.1	2005
5	Mining energy-greedy API usage patterns in Android apps: an empirical study	53	26.5	2014
6	MAPO: Mining and Recommending API Usage Patterns	185	26.4	2009
7	Semantics-based code search	169	24.1	2009
8	Live API documentation	48	24	2014
9	Portfolio: finding relevant functions and their usage	117	23.4	2011
10	Mining API patterns as partial orders from source code: from usage scenarios to specifications	205	22.8	2007

从表 4 可以看出,排在第一位的是文献“Parseweb: a programmer assistant for reusing open source code on the web”^[5]。尽管该文献的引用次数并非最多,但是在考虑了文献的发表年代后,该文献的平均每年被引次数是最多的;排在第二位的是文献“Jungloid mining: helping to navigate the API

jungle”^[41],其拥有最多的被引次数。值得注意的是,排在第五和第八位的两篇文献尽管是近两年发表的,但是却有相对较高的引用率,这是因为这两篇文献要么致力于当前流行的移动应用开发挖掘能量相关的 APIs^[8],要么采用了更好的方法来丰富 API 文档^[20]。

4.3.2 作者影响力分析

如上所述,对于每位涉及到的作者,我们计算他们的领域 H 因子来度量其在本研究领域的影响力。跟据领域 H 因子从大到小排序,表 5 列出了影响力较高的前 10 位作者。可以看到,排在第一位的是 Denys Poshyvanyk,排在第二和第三的作者分别是 Mark Grechanik 和 Tao Xie。

表 5 前 10 位高影响力作者

排名	作者	领域 H 因子	论文数	总被引数
1	Denys Poshyvanyk	9	10	392
2	Mark Grechanik	8	8	305
3	Tao Xie	7	7	836
4	SushilBajracharya	7	8	429
5	Joel Ossher	7	8	303
6	Collin McMillan	7	9	323
7	David Lo	6	8	95
8	Cristina Lopes	5	6	373
9	Susan Elliott Sim	5	5	139
10	Otavio Augusto Lazzarini Lemos	5	7	143

4.4 关键词分析

通过分析某篇文献中提到的关键词,可以基本了解该文献的研究主题。同时,分析某个研究领域相关文献中提到的所有关键词的频率,可以基本了解该领域的主要研究主题。为了获取相关文献中关键词的频率,需要对关键词进行以下预处理操作:首先,统一关键词中单词的单复数形式。因为关键词中的名词常用复数形式,预处理操作把同一个关键词中名词单复数不一致的词手动变为复数形式(如单词 software library)将变成 software libraries;其次,统一近义的关键词,例如,关键词 code search 和 source code search 将被统一为 source code search;最后,去掉某些属于该研究领域的停用词,例如 software engineering, programming 等。

经过以上步骤后,可以统计每个关键词在所有相关文献中出现的次数。表 6 列出了出现次数最多的前 10 个关键词。

表 6 前 10 个出现频率较高的关键词

排名	关键词	频率
1	source code search	56
2	Java	28
3	search engines	28
4	software reuse	25
5	software maintenance	19
6	application program interfaces (API)	18
7	data mining	16
8	information retrieval	15
9	internet	15
10	Source code	15

从表 6 可以看到,最频繁出现的词是 source code search,其在 56 篇文献中出现过,说明在该研究领域中,关于代码搜索的研究是较为主流的研究方向,这也是本文的研究重点。排在第二和第三的关键词是 Java 和 search engines,这两个关键词出现频次较多的原因可能是该领域的多数研究中使用的数据库是由 Java 语言编写的;另外,关键词 search engines

表示代码搜索的实现形式一般主要是以搜索引擎的形式存在的,这种形式便于开发者的使用。关键词 software reuse 表明该领域的研究主要是为了便于软件的重用,即使用代码搜索算法找到与查询相关的代码片段后,开发者采用复制、粘贴和修改的方式实现代码的重用。关键词 data mining 和 information retrieval 的频繁使用表明,该领域的研究主要依赖数据挖掘和信息检索的相关技术来解决代码搜索的问题。

5 网络构建与分析

本节首先介绍 GN 算法,该算法在本文中被用于寻找 3 个网络中隐藏的社区,这 3 个网络分别是作者合著网络、关键词共现网络、作者共关键词网络;然后呈现 3 个网络经分析后得到的结果。

5.1 GN 算法

GN (Girvan-Newman) 算法^[39]是一种递归算法,被用于分析隐藏在社交网络中的社区。它的主要思想是通过逐步地移除网络中隐藏的社区间的连接线,来使这些隐藏的社区呈现出来。算法 1 给出了 GN 算法的伪代码,算法具体的流程是,首先计算原始网络中每条边的介数值 (betweenness score),然后将具有最大介数值的边删去,计算所得社群结构的模块度值 (Modularity Q),再重新求解每条边对应的新的介数值,依此循环。最终,当网络中的边都被删除后,模块度值最大时的社群结构就是 GN 算法找出的最优的社群。其中,介数用于评估一条边在网络中的重要性,定义为途径该边的所有互相连接的顶点的最短路径的数量。经研究分析^[42],计算所有边的介数值的时间复杂度是 $O(mn)$,其中 m 是图中边的数量, n 是节点的数量。

算法 1 网络中的社区发现算法

Input: A relationship network $N=(A, E, W)$, A is authors set, E is edges set, W is weights set
Output: A community structure underlying the original network, which satisfying the biggest Modularity Q
Calculate the betweenness score for each edge in the original network;
While ($E \neq \text{null}$) {
 Remove the edge with biggest edge value;
 Calculate the Modularity Q , and record the network;
 Recalculate the edge values for the network after removing;}
}

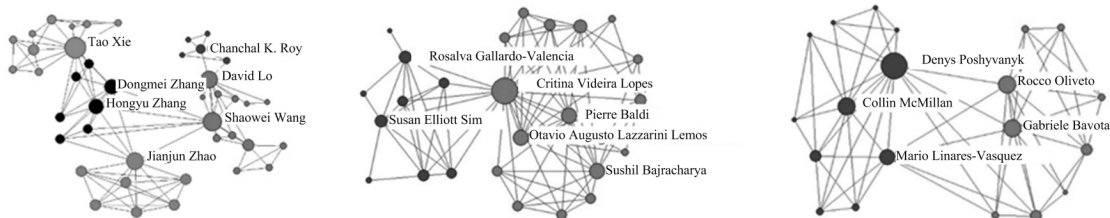


图 4 作者共著网络中前 3 个规模较大的连通子图

通过阅读每个连通图中具有最多合作次数的作者发表的相关文献,基本能了解该社区的作者合作研究的内容,例如在图 4 第一个子图中,以 Tao Xie 为代表的研究者们主要以推荐 API 作为研究内容。在图 4 第二个子图中,以 Cristina Videira Lopes 为代表的研究者们主要集中于以结构化的信息或测试用例作为输入的代码搜索研究。另外,图 4 第三个子图中以 Denys Poshyvanyk 为代表的研究者们主要集中于以自

Select the community structure with the biggest Modularity Q as the final result.

另外,模块度值 (Modularity Q) 能够体现网络划分为社群后社群结构的质量,该值逼近于 1 则说明社群结构越明显,该值逼近于 0 则说明社群结构不明显^[42],即模块度值高则代表了该算法较优,该值的范围一般为 0.3~0.7。模块度值遵循以下想法:如果某子图中边的数量超过相同子图在空模型中可能具有的内部边的预期数量,则该子图是社区。这里的空模型是原始子图的一个副本,但其中只保留原始子图的节点,而节点之间的边随机产生。模块度值的计算公式如下^[42]:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j) \quad (2)$$

其中, A 是节点之间的邻接矩阵, m 是图中边的总数, p_{ij} 表示空模型中顶点 i 和 j 之间的期望边数。如果顶点 i 和 j 在同一社区,函数取值为 1,否则为 0。

对于算法分析产生的不同社区,使用不同的颜色来标记它们;同时,使用不同大小的点来区分节点度的多少,即连接该节点的其他节点的个数。值得注意的是,一个网络可能包含一个或多个连通子图,一个连通子图可能包含多个社区,社区中的节点间具有较高的关联度。

5.2 作者合著网络

通过分析作者合著网络,能发现作者之间协作关系的一些线索。作者合著网络中包含了 93 个连通子图,这些连通子图包含了不同数目的节点,每个节点表示一个作者,作者之间的连线表示两个作者曾经共同发表过文章。图 4 显示了前 3 个规模最大的连通子图。在具有最多节点的连通子图中(最左侧),有标注了不同颜色深度的 5 个社区。来自社区内部的作者之间具有较为频繁的合作。从图 4 中可以看到,作者 Tao Xie 拥有最多的合作者,他总共与 14 人合作过。同样,在第二和第三大连通子图中,作者 Cristina Videira Lopes 和 Denys Poshyvanyk 也分别拥有最多的合作者,其合作者数量分别是 21 和 14。这里要注意的是,某个作者的合作次数并不代表该作者发表文章的数目。例如,如果某篇文章涉及到多名作者 a 、 b 和 c ,那么作者 a 就分别与作者 b 和作者 c 有合作关系。尽管作者 a 只发表了一篇文章,但是他的合作次数为 2。

由文本作为输入的代码搜索研究。

5.3 关键词共现网络

鉴于一篇文章中出现的几个关键词一般围绕同一个主题展开,通过分析关键词共现网络,能够发现文献中的研究主题。为了准确分析文献的研究主题,只考虑共同出现 3 次及 3 次以上的关键词。这是因为当两个关键词共同出现次数较少时,这两个关键词共同指向某个主题的可能性也相对较小。

图 5 显示了关键词共现网络,其包含了 869 个节点,14 个社区。包含在每个社区中的关键词共同表达一个相关的研究主题。

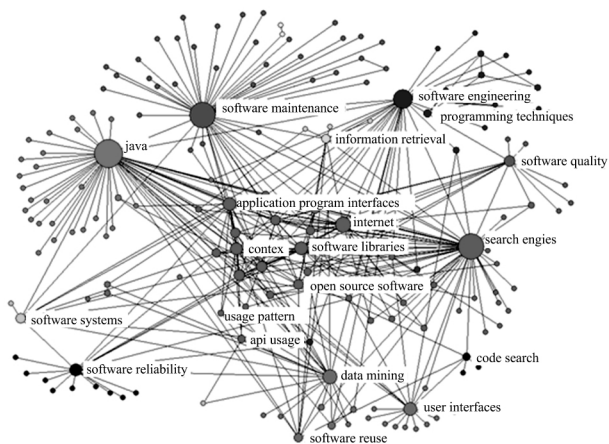


图 5 关键词共现网络包含的连通子图及社区

仔细观察图 5,有以下发现:首先可以看到各个社区之间的节点紧密交织在一起,这说明该研究领域的各个研究主题是相互关联的;其次,图中中间部分标注的社区规模相对较大,其中包含了 search engines, internet, software libraries, application program interfaces 等关键词。这是因为该社区所代表的研究是该领域的核心研究内容,即提升代码搜索和 API 推荐的性能。另外,规模较大的其他几个社区的关键词所表达的研究热点分别是软件维护 (software maintenance) 和软件可靠性 (software reliability) 等,代表了代码搜索在软件工程其他任务中的应用研究。同时,还可以看到一些较小的社区围绕在大的社区周围,代表节点有 data mining, information retrieval 等,这类节点所在社区代表了在代码搜索和 API 推荐中使用的工具和相关技术。这类社区一般较小,说明该领域的研究重点并非改进信息检索或机器学习方法本身,而是利用这些技术来提升代码搜索算法的性能。

5.4 作者共关键词网络

通过构建作者共关键词网络,可以探究作者之间共同的研究兴趣。如果两个作者使用相同关键词的数量越多,这两个作者拥有相同研究兴趣的可能性也就越大。在构建作者共关键词网络时,只考虑两个作者使用相同关键词超过 10 次的情况,这是因为,如果两个作者使用过的相同关键词较少,这两个作者拥有相同研究兴趣的概率也就相对较低。

图 6 展示了作者共关键词网络及其包含的 34 个社区。在每个社区中包含的作者具有相同的研究兴趣。

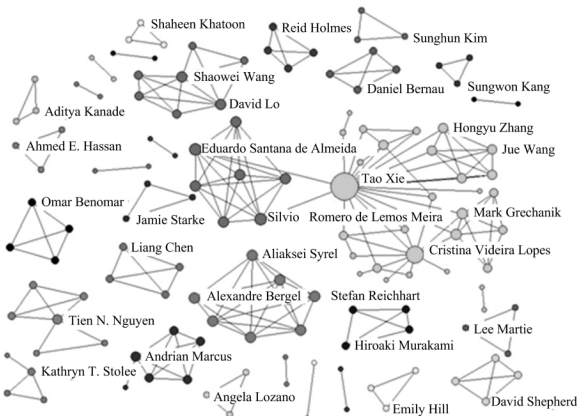


图 6 作者共关键词网络中包含的连通子图及社区

从图 6 中可以看到,最大社区由 Tao Xie 主导。此外,该社区还包括 Cristina Videira Lopes, Mark Grechanik 等作者,这些作者大多为高产作者,通过阅读这些作者发表的相关论文,可以发现他们的主要研究兴趣包括 API 使用模式挖掘、代码搜索等,该社区可以被看成是该领域的主导社区。另外,该网络中还包含了其他孤立社区,通过查看某社区主导者的相应论文,研究者能够了解该社区的研究内容。

结束语 本文借助经典的文献分析框架为代码搜索和 API 推荐领域构建了文献分析研究,致力于为研究者提供该领域的一些基本的领域知识。本文包括了 3 方面的内容:首先,识别相关文献并收集文献数据;其次,根据文献数据构建了基础文献分析,内容包括历年文献发表情况,高产的作者、机构、国家,最有影响力的文献及作者等;最后,通过构建和分析 3 个网络,分析作者之间的合作关系、文献中的研究主题、作者之间共同的研究兴趣等,进而呈现该领域最活跃的研究者以及流行的研究主题等。未来除了会不断更新该领域的文献数据,还会对该领域所蕴含的领域知识进行更进一步的挖掘,分析研究社区所发表文献的研究内容,并给出综述性的展示。

参考文献

- [1] ROBILLARD MARTIN P, WALLKER ROBERT J, ZIMMERMANN T. Recommendation systems for software engineering [J]. Software, IEEE, 2010, 27: 80-86.
- [2] KHATOON S, MAHMOOD A, LI G. An evaluation of source code mining techniques [C] // 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). 2011: 1929-1933.
- [3] XU X J, WANG W, LIU Y, et al. A Bibliographic Analysis and Collaboration Patterns of IEEE Transactions on Intelligent Transportation Systems Between 2000 and 2015 [J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 17 (8): 2238-2247.
- [4] BAJRACHARYA S K, OSSHER J, LOPES C V. Leveraging usage similarity for effective retrieval of examples in code repositories [C] // Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2010: 157-166.
- [5] THUMMALAPENTA S, XIE T. Parseweb: a programmer assistant for reusing open source code on the Web [C] // Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering. Atlanta, Georgia, USA, 2007.
- [6] NEWMAN M E J. Modularity and community structure in networks [C] // Proceedings of the National Academy of Sciences. 2006: 8577-8582.
- [7] LAZZARINI L O A, KRISHNA B S, JOEL O, et al. CodeGenie: using test-cases to search and reuse source code [C] // Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering. Atlanta, Georgia, USA, 2007.
- [8] LINARES-VÁSQUEZ M, BAVOTA G, BERNAL-CÁRDENAS C, et al. Mining energy-greedy API usage patterns in Android apps: an empirical study [C] // Proceedings of the 11th Working Conference on Mining Software Repositories. 2014: 2-11.

- [9] ZHONG H, XIE T, ZHANG L, et al. MAPO: Mining and recommending API usage patterns[C]//European Conference on Object-Oriented Programming. 2009;318-343.
- [10] MCMILLAN C, POSHYVANYK D, GRECHANIK M, et al. Portfolio: Searching for relevant functions and their usages in millions of lines of code[C]//ACM Transactions on Software Engineering and Methodology (TOSEM). 2013;37.
- [11] KEIVANLOO I, RILLING J, ZOU Y. Spotting working code examples[C]//Proceedings of the 36th International Conference on Software Engineering (ICSE). Hyderabad, India, 2014.
- [12] RAGHOTHAMAN M, WEI Y, HAMADI Y. SWIM: Synthesizing What I Mean[J]. arXiv preprint arXiv:1511.08497, 2015.
- [13] JIANG H, NIE L M, SUN Z Y, et al. ROSF: Leveraging Information Retrieval and Supervised Learning for Recommending Code Snippets[OL]. <http://doi.ieeecomputersociety.org/10.1109/TSC.2016.2592909>.
- [14] LI X, WANG Z R, WANG Q X, et al. Relationship-Aware Code Search for JavaScript Frameworks[C]//Proceedings of the 24th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2016). Eattle, WA, 2016.
- [15] WANG S W, LO D, JIANG L X. Active code search: incorporating user feedback to improve code search relevance[C]//Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. Vasteras, Sweden, 2014.
- [16] LU M L, SUN X B, WANG S W, et al. Query expansion via WordNet for effective code search[C]//2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER). 2015;545-549.
- [17] LEMOS O A L, DE PAULA A C, ZANICHELLI F C, et al. Thesaurus-based automatic query expansion for interface-driven code search[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. 2014;212-221.
- [18] NIE L M, JIANG H, REN Z L, et al. Query Expansion Based on Crowd Knowledge for Code Search[J]. IEEE Transactions on Services Computing, 2016, 9(5):771-783.
- [19] LV F, ZHANG H Y, LOU J G, et al. CodeHow: Effective Code Search Based on API Understanding and Extended Boolean Model (E) [C] // 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2015;260-270.
- [20] SUBRAMANIAN S, INOZEMTSEVA L, HOLMES R. Live API documentation[C]//Proceedings of the 36th International Conference on Software Engineering. 2014;643-652.
- [21] MORENO L, BAVOTA G, DI PENTA M, et al. How Can I Use This Method? [C]//2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE). 2015;880-890.
- [22] LEMOS O A L, BAJRACHARYA S, OSSHER J, et al. A test-driven approach to code search and its application to the reuse of auxiliary functionality [OL]. <http://www.zentralblatt-math.org/ioport/en/search/?q=an%3A05854508>.
- [23] STOLEE K T, ELBAUM S, DOBOS D. Solving the search for source code[C]//ACM Transactions on Software Engineering and Methodology (TOSEM). 2014;26.
- [24] INOUE K, SASAKI Y, XIA P, et al. Where Does This Code Come from and Where Does It Go? Integrated Code History Tracker for Open Source Systems[C]//2012 34th International Conference on Software Engineering (ICSE). 2012;331-341.
- [25] THUNG F, WANG S W, DAVID L, et al. Automatic recommendation of api methods from feature requests [C] // 2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE). 2013;290-300.
- [26] GU X D, ZHANG H Y, ZHANG D M, et al. Deep API Learning[J]. arXiv preprint arXiv:1605.08535.
- [27] NIU H R, KEIVANLOO I, ZOU Y. Learning to rank code examples for code search engines[OL]. <http://post.queensu.ca/~zouy/files/EMSE-Haoran-2015.pdf>.
- [28] RAHMAN M M, ROY C K, LO D. RACK: Automatic API Recommendation Using Crowdsourced Knowledge [C] // 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). 2016;349-359.
- [29] VAN NOORDEN R, BRENDAN M, REGINA N. The top 100 papers[J]. Nature, 2014, 514(7524):550-553.
- [30] HEE P D, KYEONG K H, YOUNG C I, et al. A literature review and classification of recommender systems research[J]. Expert Systems with Applications, 2012, 39(11):10059-10072.
- [31] VAHID G, MÄNTYLÄ MIKA V. Citations, research topics and active countries in software engineering: A bibliometrics study [J]. Computer Science Review, 2016, 19:56-77.
- [32] CLAES W. An analysis of the most cited articles in software engineering journals-2001[J]. Information and Software Technology, 2008, 50(1/2):3-9.
- [33] CAI K Y, CARD D. An analysis of research topics in software engineering-2006[J]. Journal of Systems and Software, 2008, 81(6):1051-1058.
- [34] FERNANDES J M. Authorship trends in software engineering [J]. Scientometrics, 2014, 101(1):257-271.
- [35] LU Z B. How are excellent (highly cited) papers defined in bibliometrics? A quantitative analysis of the literature [J]. Research Evaluation, 2014, 23(23):166-173.
- [36] AFZAL W, TORKAR R, FELDT R. A systematic review of search-based testing for non-functional system properties[J]. Information and Software Technology, 2009, 51(6):957-976.
- [37] SERENKO A, BONTIS N. Meta-review of knowledge management and intellectual capital literature: citation impact and research productivity rankings[J]. Knowledge & Process Management, 2004, 11(3):185-198.
- [38] BATISTA P D, CAMPITELI M G, KINOCHI O, et al. An index to quantify an individual's scientific research valid across disciplines[J]. Eprint Arxiv Physics, 2005, 68(1):179-189.
- [39] GIRVAN M, NEWMAN M E J. Community Structure in Social and Biological Networks[J]. Proceedings of the National Academy of Sciences of the United States of America, 2002, 99(12):7821-7826.
- [40] BORGATTI S P, EVERETT M G, FREEMAN L C. UCINET VI for windows: Software for social network analysis[OL]. http://pages.uoregon.edu/vburris/hc431/Uciet_Guide.pdf.
- [41] MANDELIN D, XU L, BODODÍ K R, et al. Jungloid mining: helping to navigate the API jungle[J]. ACM SIGPLAN Notices, 2005, 40(6):48-61.
- [42] CLAUSET A, NEWMAN M E J, MOORE C. Finding community structure in very large networks [J]. Physical Review E, 2004, 70(6pt2):264-277.