2-2017

检测GitHub上的类似存储库

张云

大卫-罗
新加坡管理大学，davidlo@smu.edu.sg

pavneet singh kochhar
新加坡管理大学，kochharps.2012@phdis.smu.edu.sg

夏新 李全

来

其他作者见下页

作者
张云，罗大伟，PAVNEET SINGH KOCHHAR，夏新，李全来，孙建玲

SANER 2017。第24届IEEE国际软件分析、进化和再造会议论文集。克拉根福。
2017年2月20日至24日，奥地利，http://doi.org/10.1109/SANER.2017.7884605

检测GitHub上的类似存储库

Quanlai Li[1] , Yan Li[1] , Pavneet Singh Kochhar[2] , Xin Xia[1] and David Lo [21]
浙江大学计算机科学与技术学院，中国杭州[2]
新加坡管理大学信息系统学院，新加坡
{lql, liyansam, xxia}@zju.edu.cn, {kochharps.2012, davidlo}@smu.edu.sg.

摘要--
***GitHub*包含数以百万计的存储库，其中有许多实现类似功能的存储库。在GitHub上寻找类似的资源库对软件工程师来说很有帮助，因为它可以帮助他们重用源代码，识别改变原生的实现，探索相关的项目，找到可以贡献的项目，并发现代码盗窃和剽窃行为。以前的研究提出了通过分析API使用模式和软件标签来检测类似应用程序的技术。不幸的是，这些先前的研究要么只利用了有限的信息源，要么使用了GitHub上的项目所没有的信息。**

在本文中，我们提出了一种能够有效地检测GitHub上类似仓库的方法。我们的方法是基于三个启发式方法设计的，这三个启发式方法利用了以前的工作中没有考虑到的额外数据源（即GitHub星级和*readme*文件）。这三个启发式方法是：在很短的时间内被相同的用户*加星的*项目有可能彼此相似，被相似的用户加星的项目有可能彼此相似，而其readme文件包含相似内容的项目有可能彼此相似。基于这三个启发式方法，我们计算出两个相关性分数（即*基于星级的相关性*和*基于读物的相关性*）来评估两个资源库之间的相似度。通过整合这两个相关性分数，我们建立了一个名为*RepoPal*的推荐系统来检测类似的资源库。我们使用GitHub上的一千个Java资源库，将*RepoPal*与之前最先进的方法*CLAN*进行比较。我们的实证评估表明，*RepoPal*比*CLAN*取得了更高的成功率、优惠和信心。

关键词--*相似*库，GitHub，信息再三性，推荐系统

I. 简介

GitHub包含超过2900万个仓库[1]，这些仓库由分布在世界各地的1100多万名开发者开发。仓库是GitHub的一个基本单元，通常包含软件项目的源代码和资源文件。它还存储了与项目的演变历史和高级功能有关的信息，以及创建、贡献、分叉、启动和观察项目的人员。GitHub承载了无数的项目，包括数据库应用、操作系统、游戏软件、网络小程序、移动应用等等。像谷歌、Facebook和微软这样的大型组织正在使用GitHub来托管他们的开源项目。许多有影响力的开源项目也转移到GitHub上；这些项目包括流行的编程语言项目，如Python[2] 和Go[3] ，流行的服务器项目，如Nginx[4] 和Cherokee[5] ，流行的开发框架、平台和库，如Bootstrap[6] ，Node.js[7] ，和JQuery[8] ，等等。大量的项目给开发者提供了大量的选择，可以选择他们想使用或贡献的项目。

在GitHub承载的数百万个仓库中，有许多实现了类似的功能，但由不同的开发者或组织开发。检测这些类似的仓库对于代码重用、快速原型、识别替代性实现、探索相关项目、寻找可贡献的项目、发现代码盗窃和剽窃（当它们被不适当地重用时）等等都很有用--例如，[1]、[2]、[3]。一项研究表明，经常有超过50%的源代码文件在一个以上的开源项目中被重复使用[4]。因此，通过检测类似的应用程序，开发人员可以重用代码，并专注于实现任何现有应用程序所不提供的功能。

遗憾的是，就我们所知，目前GitHub上还没有实现任何系统可以检测到仓库的相似性。为了帮助开发者在数以百万计的软件库中找到相关的软件库，GitHub提供了一个搜索引擎。然而，它只是一个简单的基于文本的搜索引擎，接受查询词列表作为输入，并返回包含查询词的仓库名称、文件、问题报告和用户名称。这个搜索引擎当然不是一个寻找类似仓库的理想工具。

在文献中，过去的研究已经提出了几种检测类似应用程序的技术。McMillan等人开发了一种名为*CLAN*（Closely reLated ApplicatioNs）的方法，通过比较两个应用程序的API调用来计算Java应用程序之间的相似性[5]。他们表明，开发人员能够找到类似的应用程序，他们的技术比之前提出的另一种技术MUDABlue[6]表现得更好。Thung等人提出了一种技术，利用软件标签而不是*CLAN*使用的API使用模式，来推荐类似的应用程序[7]。他们的方法通过给不同的标签分配不同的权重来自动识别一个应用程序使用的重要标签。然后，这些标签和它们的权重被用来比较应用程序。

[1]https://github.com/about/press [2]https://github.com/python
[3]https://github.com/golang
[4]https://github.com/nginx/nginx
[5]https://github.com/cherokee/webserver [6]https://github.com/twbs/bootstrap
[7]https://nodejs.org/en/
[8]https://github.com/jquery/jquery

尽管之前的工作在识别类似应用方面取得了重大进展，但将其应用于寻找GitHub上的类似存储库仍有不少挑战。首先，GitHub包含数以百万计的存储库，这些存储库随着时间的推移而频繁更新，定期静态分析它们以检索API调用是一项昂贵的任务。其次，GitHub不允许用户对存储库进行标记。除了这些挑战，之前的工作没有利用GitHub特有的额外数据源来提供新的洞察力。例如，GitHub允许用户给仓库*加星*，以跟踪他们认为有趣的仓库。给仓库加星是一个公开的活动，可以被其他人查看和追踪。此外，软件库通常包含描述软件库中开发的应用程序的高级功能的自述文件。

为了处理这些限制，并利用额外的数据源，在这项工作中，我们提出了一种新的方法来识别GitHub上的类似存储库。我们的工作是基于三个启发式方法。首先，在很短的时间内被同一个用户加了星的仓库很可能是相似的。第二，被相似的用户加星的存储库很可能是相似的。第三，那些自述文件有相似内容的软件库很可能是相似的。基于这三个启发式方法，我们计算了两个相关性分数，以衡量两个资源库的相似程度。第一个相关性分数，称为*基于星的相关性*，是基于前两个启发式方法。它是通过以下方式计算的。(1)计算给两个资源库加星的人数，根据每个人给两个资源库加星的时间间隔进行加权，以及(2)计算给资源库加星的类似用户对的数量。第二个相关性分数，被命名为*基于读取的相关性*，是基于第三个启发式方法的。它是通过计算两个存储库的自述文件的向量空间表示的余弦相似度来计算的。我们提出的方法，名为***RepoPal***，结合了这两个相关性分数来识别给定目标库的类似库。

我们评估了***RepoPal***从GitHub的1,000个Java仓库中推荐类似仓库的能力。我们总共使用了50个查询，每个查询对应GitHub中的一个Java仓库，以评估***RepoPal***和最先进的方法***CLAN***[5]的有效性。由于***CLAN***只能处理Java程序，所以我们只关注Java仓库。我们通过一个用户研究来评估***RepoPal***和***CLAN***的有效性。参加研究的人对每个推荐的类似项目进行评分，评分范围从1（高度不相关）到5（高度相关）。根据评分，我们用三个标准来衡量有效性：成功率[7]、可信度[5]、[7]和精确度[5]、[7]作为衡量标准[9]。我们的用户研究结果显示，***RepoPal***在成功率、置信度和精确度方面分别比***CLAN***高出34.48%、20.14%和41.03%。

我们工作的贡献如下。

- 我们提出了三个新的启发式方法来识别GitHub上的类似仓库，这些方法利用了之前工作中没有考虑的两个数据源--
  即GitHub星级和自述文件。我们将这三个启发式方法整合到一个工具***RepoPal***中。
- 我们在1000个Java资源库的数据集上对***RepoPal***和***CLAN***进行了评估，结果显示我们的技术在成功率、置信度和精确度方面都大大优于***CLAN***。

本文其余部分的结构如下。在第二节中，我们描述了我们的方法所使用的三种启发式方法和一些激励性的例子。在第三节中，我们阐述了我们的方法***RepoPal***的细节。在第四节中，我们描述了实验的设置，将***RepoPal***应用于1000个GitHub项目的数据集，并与最先进的方法***CLAN***进行了比较。我们在第五节和第六节分别介绍了我们的实验结果和对有效性的一些威胁。第七节简要回顾了相关工作。第八节是结论，并提到了未来的工作。

II. 启发式方法

在这一节中，我们描述了我们系统中使用的三种启发式方法，并在第II-A、II-B和II-C节中通过一些例子来说明它们。

*A. 启发式1：在短时间内被同一用户*评为明星的*项目，很可能彼此相似。*

GitHub用户可以给一个仓库加星，以显示他们的认可和兴趣。给GitHub仓库加星的用户被称为该仓库的*追星族*。追星者可以持续获得被标记的仓库的最新信息。一个GitHub用户在短时间内给多个仓库加星的事实可能表明这些仓库是相似的。

当GitHub用户觉得某个仓库有趣或有用时，他们往往有足够的动力将其升星。如果一个仓库中开发的代码被一个追星族使用，她可能想跟踪它，以便在原始仓库更新时更新自己的代码。即使开发者不使用版本库中的代码，她仍然可以给一个有趣的版本库加星，让她更容易访问它，并有可能在将来使用版本库中开发的代码。由于加星是GitHub的一项公共活动，给一个仓库加星表明加星者对该仓库的赞赏和认可。评星可以作为推广感兴趣的仓库的一种手段；它会出现在追星者的公共活动时间轴上，让追星者的追随者知道该仓库的情况。GitHub本身也鼓励用户给仓库加星。仓库的星数被用于GitHub的许多功能中。例如，GitHub使用各种仓库的星数来帮助其搜索引擎返回的仓库排名。[10]GitHub还通过把那些快速积累星级的仓库放在*GitHub探索*页面上来推广它们。[11]

当一个软件开发者遇到问题时，他可能会向GitHub寻求帮助，希望能找到一些代码重用和灵感的存储库。在GitHub上冲浪的过程中，开发者可能会遇到一些有用的仓库，并随后将那些与问题相关的仓库加星。这样一来，同一个GitHub用户就可以在短时间内将多个相关的软件库加星。该问题

9这些评价指标的定义在第四节给出。

10https://help.github.com/articles/about-stars/ 11https://github.com/explore

开发人员遇到的问题经常随着时间的推移而变化。直观地说，单个开发人员遇到的问题往往是相似的。如果遇到问题的时间间隔很短，那就更是如此。一个开发者可能在上午解决一个问题，下午解决另一个相关的问题，5天后解决一个不太相关的问题。我们的第一个启发式方法是基于这样的假设：一个用户在较短的时间段内盯住的资源库可能比该用户在较长的时间段内盯住的资源库有更高的相似度。

例如，考虑GitHub用户LiqiangZhang[12]和他在2015年11月18日的公开活动。在那一天，他在一个小时内对三个仓库进行了加星，即bunnyblue/DroidFix、jasonross/Nuwa和dodola/HotFix。这些软件库都与安卓热修复有关。[13]如果我们看一下他两天前的活动，我们会注意到他也在一小时内给三个软件库加了星，即spongebo-brf/MaterialIntroTutorial，alafighting/CharacterPickerView，和fengjundev/DoubanMovie-React-Native。这三个资源库都是Android设计和用户交互项目。11月18日上星的软件库彼此高度相似，而11月16日上星的软件库也彼此高度相似。这两组软件库的相似度较低，但却仍然相似，因为它们都是Android软件库。这个例子说明，被一个用户放在一起的软件库很可能是相似的。如果它们被放在一起的时间很短，这一点尤其正确。

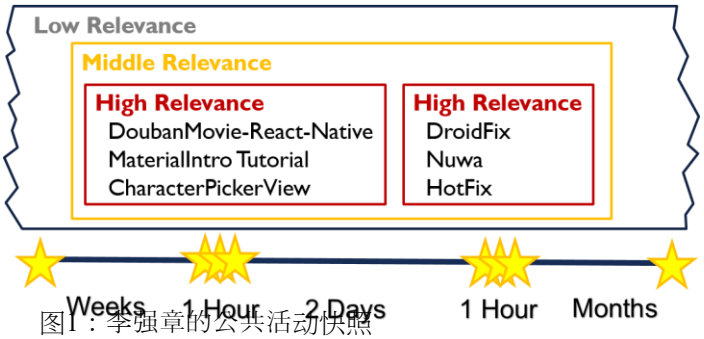


图1：李强章的公共活动快照

这种现象不仅限于李强章。而我们找到了很多类似的例子。2015年11月23日和24日，GitHub用户fenixlin[14]对两个仓库进行了加星：heshibidahe/Active-learning-ml-100k和scikit-learn/scikit-learn。它们都是Python机器学习工具或模块。2016年1月26日，GitHub用户shichaohao[15]上星了两个软件库：getlantern/lantern和ziggear/shadowsocks。一旦部署，这两个

可用于评估各地区的公共网站在这些网站被封锁的地方。另一位GitHub用户DreaminginCodeZH[16]，在2016年1月17日的时候，将三个软件库的内容进行了星级化，包括timusus/RecyclerView-FastScroll, AndroidDeveloperLB/ThreePhasesBottomSheet和daimajia/AndroidImageSlider。这些资源库的相似之处在于，它们都处理Android上的滚动或滑动动作。

此外，从直观上看，GitHub用户对同一组仓库加星的次数越多，这组仓库就越有可能彼此相似。例如，GitHub用户yangweidong[17]，在2015年11月18日的一个小时内，将bunnyblue/DroidFix和dodola/HotFix都列为明星。这两个仓库都是yangweidong和LiqianZhang的星级。GitHub用户vinci7[18]在2016年1月27日将timusus/RecyclerView-FastScroll和AndroidDeveloperLB/ThreePhasesBottomSheet都列为星级。正如我们之前提到的，这两个资源库也被用户DreamingCodeZH评为星级。许多人把两个软件库放在一起的事实，加强了我们对这两个软件库相似的信心。

B. 启发式2：被类似用户评为星级的项目很可能彼此相似。

在第一种启发式方法中，我们认为被同一用户标示的存储库是相似的。在这个启发式中，我们认为被相似的用户加了星的存储库也是相似的。我们认为GitHub用户有很多共同的星级仓库是相似的。

例如，我们追踪了两个GitHub用户，他们至少在五个共同的仓库里挂了星。这两个GitHub用户，Will Sahatdjian[19]和Aldiantoro Nugroho[20]，分别在contra/react-responsive[21]、Ramotion/folding-cell[22]、so-fancy/diff-so-fancy[23]、corymsmith/react-native-fabric[24]和danielgindi/ios-charts[25]。因此，我们认为他们是类似的用户。Will Sahatdjian和Aldiantoro Nugroho的许多资料库也很相似。例如，Will Sahatdjian将jessesquires/JSQMessagesViewController[26]，而Aldiantoro Nugroho则将facebook/pop[27]。这两个资源库都没有被其他用户加星，然而，它们是相似的。前者描述自己是*一个优雅的iOS消息UI库*，而后者描述自己是*一个可扩展的iOS和OS X动画库，对基于物理的交互很有用。*

C. 启发式3：其自述文件包含类似内容的项目很可能彼此相似。

分析readme文件的文本相似性也可以帮助我们找到类似的存储库。直观地说，具有类似功能的资源库在其readme文件中使用类似词语的可能性更高，即使它们不是由同一组开发人员或组织开发的。

[12]https://github.com/StormGens

13Android热修复允许开发者在不发布新版本的情况下更新Android应用程序。

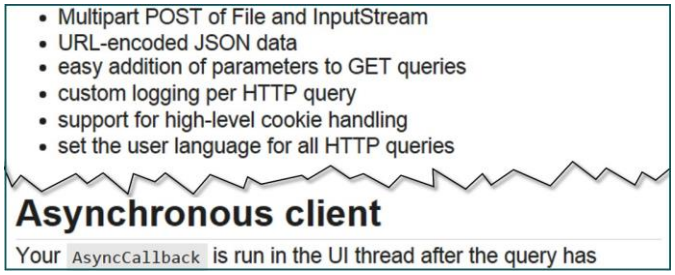
[14]https://github.com/fenixlin

[15]https://github.com/shichaohao

[16]https://github.com/DreaminginCodeZH

17 https://github.com/yangweidong
18 https://github.com/vinci7
19 https://github.com/kwcto?tab=activity
20 https://github.com/kriwil
21 https://github.com/contra/react-responsive
22 https://github.com/Ramotion/folding-cell
23 https://github.com/so-fancy/diff-so-fancy
24 https://github.com/corymsmith/react-native-fabric
25 https://github.com/danielgindi/ios-charts
26 https://github.com/jessesquires/JSQMessagesViewController 27 https://github.com/facebook/pop

例如，考虑两个资源库Android- HttpClient[28] 和 android-async-http[29]
。这两个资源库彼此相似，因为它们实现了一些共同的功能，例如，Android应用程序的异步HTTP客户端功能。图2显示了它们的readme文件的摘录，我们发现它们共享一些词汇，例如，异步、HTTP、cookie、JSON、GET、POST等。



(a) Android-HttpClient的自述文件中的一个例子

(b) 来自android-async-http自述文件的一个例子

图2：两个类似项目的自述文件

III. 报道

在这一节中，我们首先解释 "中国 "的整体架构。*RepoPal*。接下来，我们介绍它的每一个主要组成部分。

A. 建筑学

图3显示了我们的技术*RepoPal*的整体架构，包括其组成部件、输入和输出。它的输入是一组GitHub仓库（GitHub Repository
Set），和一个查询库（Query
Repository）。它输出一个与查询相似的存储库的排序列表
储存库（Similar Repository List）。它由三个主要部分组成。Star Relevance Calculator, Readme Relevance Calculator, and Composer.第一个和第二个组件
分别计算基于星的相关性分数和基于读的相关性分数。最后一个组件将这两个相关性分数结合起来，根据它们与查询库的相似性对库集中的库进行排名。我们在接下来的小节中描述*RepoPal*的三个组件。

B. 星级关联度计算器

星级相关性计算器组件利用GitHub的星级来对仓库进行排名。直观地说，两个仓库在相似的时间段内被许多人加了星号

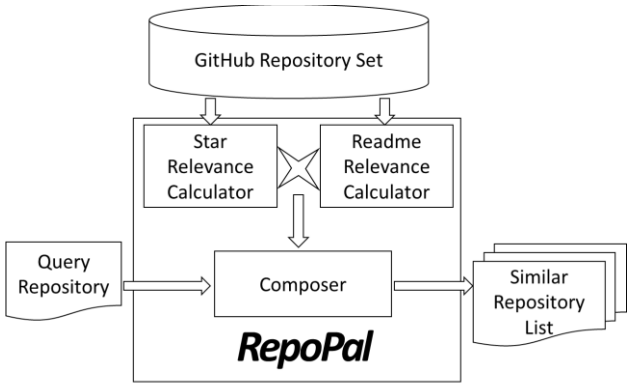


图3：*RepoPal*架构

有可能在一起是相似的。我们已经在第二节中展示了一个激励性的例子。我们使用这个直觉来计算两个资源库之间的*星型相关分数*。

在我们定义一个公式来计算基于星星的相关性分数之前，我们需要介绍几个符号。让U表示一个GitHub用户，R表示一个仓库，$S(R)$表示所有给R加星的用户。考虑到用户U给仓库R加星的事实，我们用$T(U, R)$表示用户U给仓库R加星的时间戳。给出两个被用户U*评为星级的*资源库R_1和R_2，我们可以计算出它们被U评为星级的*调整时间差*（表示为$D(U, R_1, R_2)$），如下所示。

$$D(U, R_1, R_2) = |T(U, R_1) - T(U, R_2)| + 30min \quad (1)$$

我们将调整后的时差设定为实际时差（分钟）加30分钟。这是因为，无论两个资源库是在一分钟内还是在几分钟内上星，它们之间的相关性都没有太大的区别。

基于调整后的时间差，我们可以计算两个资源库R_1和R_2之间的基于星的相关性分数，表示为$Relevance_s(R_1, R_2)$，如下所示。

$$\text{相关性}_s(R_1, R_2) = \sum_{U \in S(R_1) \cup S(R_2)} \frac{1}{D(U, R_1, R_2)^2} + E_{(U_i, U_j) \in SIM} \quad (2)$$

28https://github.com/levelup/Android-HttpClient
29https://github.com/loopj/android-async-http

在上述公式中，SIM是一组相似的GitHub用户对。我们认为一对GitHub用户给5个共同的存储库加星是相似的。同时，E是一个小数，等于U_i 和U_j的任何一对存储库的最大调整时间差的倒数。设置E为这样的小数的动机是，相似用户的两个存储库的相似度不应该大于同一用户的两个存储库的相似度。

基于星级的相关性分数被计算为调整后的时间差的倒数加上类似用户的一些贡献之和。这样一来，对于两个资源库来说，当它们两个星之间的时间差较小时，倒数就会变大，因此，相关性是

更高。当有更多的用户将这两个资源库放在一起，相关性分数也会更高。当有更多相似的用户给这两个资源库加星时，相关性分数也会更高。

C. Readme 相关性计算器

Readme Relevance Calculator组件通过比较两个资源库的readme文件来计算基于readme的相关性分数。我们使用矢量空间模型（VSM）来计算这个相关性分数，该模型通常用于在信息检索中寻找文档之间的相似性。我们对所有的自述文件进行预处理，去掉止损词，并进行词根处理，将单词减少到其词根形式。然后我们将文件转换成权重向量。每个预处理过的词对应于向量中的一个元素，其权重用标准的tf-idf加权方案来计算[8]。给定文件（即readme文件）R在文件集合C中的单词t的权重，表示为$w_{t,R}$，计算方法如下。

$$w_{t,R} = (1 + \log \mathrm{tf}_{t,R}) \times \log\left(\frac{|C|}{df_t}\right) \quad (3)$$

在上面的公式3中，$tf_{t,R}$表示单词t在文档R中的词频，即t在读物文件R中出现的次数。df_t表示t的文档频率，即包含单词t的文档（即C中的读物文件）的数量。在计算出权重后，每个读物文件R可以表示为一个权重向量。给定两个资源库和它们的两个代表权重向量，我们可以通过取它们的代表向量的余弦相似度来计算它们的读物相关性得分，表示为$Relevance_r(R_1, R_2)$。[2] 如下：

$$\text{相关性}_r(R_1, R_2) = \frac{\sum_{t \in R_1 \cap R_2} w_{t,R_1} \times w_{t,R_2}}{\sqrt{\sum_{t \in R_1} w_{t,R_1}^2} \times \sqrt{\sum_{t \in R_2} w_{t,R_2}^2}} \quad (4)$$

D. 作曲家

Composer组件将两个相关性分数进行组合，并计算出两个资源库R_1和R_2的*总体相关性分数*，具体如下。

$$Relevance(R_1, R_2) = Relevance_s(R_1, R_2) \times \text{相关性}_r(R_1, R_2) \quad (5)$$

算法1中显示了Composer组件的伪代码。给定一个查询存储库，它计算查询存储库和存储库集（RepoSet）中的每个存储库之间的星型、读取型和整体相关性分数--第1-5行。然后，存储库集合中的存储库将根据它们的整体相关性分数进行排序（第6行）。最后，输出前k个存储库（第7行）。

算法1：寻找最相似的前K个储存库

输入:QRepo:查询存储库，RepoSet。存储库的集合
输出。排名前十的存储库
1 **对于***RepoSet中的每个存储库r*，**做**
2 计算*相关性*$_s$ *(QRepo, r)*
3 计算*相关性*$_r$ *(QRepo, r)*
4 计算*相关度*= *相关度*$_r$ × *相关度*$_s$
5**结束**
6 根据*相关性*得分，将RepoSet资源库按降序排列。
7 输出前k个存储库

IV. 实验设置

我们将*RepoPal*与之前的工作*CLAN*（Closely reLated ApplicatioNs）[5]进行比较。就我们所知，*CLAN*是最相似通过测量它们的Java API（即JDK）方法调用的相似性，来确定类似的应用程序。*CLAN*解析程序的源代码，并通过它所调用的Java API方法表示每个程序。然后，*CLAN*按照流行的*术语频率-反文档频率*（TF-IDF）加权方案[8]为Java API方法分配权重；方法调用次数较多的方法被赋予较高的权重，而在较少的应用中被调用的方法被赋予较高的权重。赋予更高的权重。接下来*CLAN*使用潜在语义索引（LSI）[9]，根据加权的JDK方法调用对两个应用程序进行比较。我们还将*RepoPal*和*CLAN*结合起来，并将结合后的系统表示为*Combined*。给定一对资源库R_1和R_2，*Combined*将*RepoPal*产生的相关性分数与*CLAN*产生的分数相乘。这两个相关性分数的乘积就是*Combined*为R_1和R_2所返回的分数。

为了评估这三个系统（即*RepoPal*、*CLAN*、和*结合起来*），我们使用GitHub上1000个独特的[31]流行的Java存储库的数据集，这些存储库得到了20多颗星的GHTorrent [10][32]。这些存储库是随机挑选的，星星的数量从21到6106不等。我们设定星级数量的要求，是因为GitHub上的许多仓库质量不高[11], [12]，尤其是当它们没有很多星级的时候。理想情况下，只有高质量的应该推荐的是存储库。我们只考虑Java资源库，因为我们想把我们的方法与只适用于Java的*CLAN*作比较。对于这1,000个资源库中的每一个，我们都会收集其readme文件、star events和source代码。我们在1,000个资源库中挑选50个作为查询对象（见表一），并使用*RepoPal*、*CLAN*和*Combined*生成前五个相似的资源库。

的相关工作。[30]*CLAN*确定了

我们邀请了50位参与者来评估这三个系统（即*RepoPal*、*CLAN*和*Combined*）在推荐类似资源库方面的有效性。参与者来自阿里巴巴有限公司、网易有限公司和浙江大学。在50名参与者中，有12名是阿里巴巴员工。其中9人是网易的员工，25人是浙江大学的高年级本科生，4人是同一所大学的研究生。他们都是熟练的程序员，至少有3年的编码经验。48名参与者是GitHub的用户，其中13人经常在GitHub上搜索有趣的仓库。

每个参与者都得到了一个查询库

30 Thung等人的另一项密切相关的工作[7]不能适用于我们的设置，因为它需要有手动分配的标签，而GitHub不支持标签的使用。

31 没有一个存储库是相互克隆的。

32 http://ghtorrent.org/

表一:用来评估*RepoPal*、*CLAN*和合并的查询。

数值。	查询	数值。	查询	数值。	查询	数值。	查询	数值。	查询
1	激活	11	弹性图像视图(FlexiImageView)	21	jpropel	31	apifest-oauth20	41	FragmentTabHostExample
2	安洛兹奇	12	gatk	22	json	32	循环-进度-按钮	42	mvp-to-mvvm-transition
3	脚本解析器(cli-parser)	13	图形茶	23	jsonde	33	O2OMobile Android	43	安卓-数字-变形
4	紧缩	14	GwtMobile-UI	24	煤油灯	34	安卓-archetypes	44	lucy-xss-servlet-filter
5	me.fantouch.libs	15	悬空-UI	25	这里的抒情	35	玩耍-休息-安全	45	衡量标准-报告者-配置
6	dl4j-examples	16	高尺度-lib	26	砾石	36	多动作文本视图	46	SalesforceMobileSDK-Android
7	Dumbledroid	17	调用者	27	Pydev	37	多项选择阿尔本	47	弹簧-集成-扩展
8	凌晨时分	18	iText-4.2.0	28	齐钛	38	深度功能测试	48	aws-apigateway-swagger-importer
9	表情符号聊天	19	itl-java	29	询问	39	浏览器	49	购物清单-清洁-架构-示例
10	冯氏玻璃	20	工作机会	30	暂时性的	40	appbundle-maven-plugin	50	飞利浦HueSDK-Java-多平台-安卓版

上文中列出的是三个系统所产生的15个检索库（每个系统5个）。在某些情况下，可能有几个共同的资源库被不同的系统用一个查询检索出来，我们省略了重复的部分。为了减少实验的偏差，我们不告知参与者三个系统中的哪一个推荐了一个资源库。参与者得到了GitHub资源库的URL，因此可以访问代码、作者和贡献者、自述文件、相关链接（如果有的话）和其他信息来评估相似度。为了进一步减少偏见，我们没有指示参与者关注某个特定的信息。我们希望参与者能够利用各种信息来源公平地判断资源库的相似性。参与者被要求仔细理解所有的资料库，并至少用30分钟的时间来评估相似性。之后，他们会被问到以下问题，即每个检索到的资源库与查询的相关性。

检索到的资源库与查询的相关性如何资源库？
选项。
(1) 高度不相干。
参与者发现，在检索和查询库之间完全没有任何共同点。
(2) 无关紧要。
参与者发现这两个资源库没有什么共同之处。
(3) 中立。
参与者发现，这两个资源库的相关性不大。
(4) 相关的。
参与者发现，这两个资料库在很多方面都很相似。
(5) 高度相关。
参与者发现，检索和查询的资源库在大多数方面是相似的，甚至有些部分可能是相同的。

我们将每个参与者的反应映射为1-5分，1分对应"高度不相关"，5分对应"高度相关"。我们使用这些分数来评估三个推荐系统（即*RepoPal*、*CLAN*和*Combined*）的有效性。

按照先前的研究[5]，[7]，我们使用三个评价指标，即成功率、置信度和精确度，来总结我们从参与者那里得到的评分。

1) *SuccessRate@T*。SuccessRate@T被定义为一个系统产生的前5名推荐的成功比例。一个前五名的推荐被认为是如果至少有一个检索到的存储库的评级为T或更高，则成功。我们考虑SuccessRate@4和SuccessRate@5。
2) *信心*。信任度的中位数和平均值被定义为参与者对系统推荐的所有检索库的评价中位数和平均值。
3) *精度*。精度被定义为系统为一个查询推荐的相关和高度相关的资源库的比例。给定一组查询，我们可以定义精度分数的平均值和中位数。[33]

基于上述评价指标，我们调查了以下研究问题。

RQ1:
RepoPal、CLAN和Combined至少返回一个相关（或高度相关）搜索结果的查询比例是多少？
问题2：与CLAN相比，使用RepoPal的参与者的信心中位数和平均值有多高？
RQ3:RepoPal的精确性得分是多少？CLAN?

V. 实验结果

在这一节中，我们报告了这三个系统在三个指标上的评价，并使用从参与者那里收集的数据来检查差异的重要性。

A. RQ1：成功率

表二显示了*RepoPal*、*CLAN*和*Combined*的成功率。我们注意到，*RepoPal*和*Combined*的成功率比*CLAN*高。*RepoPal*的表现比*Combined*好，表明将*CLAN*加入*RepoPal*并不能帮助提高性能。*RepoPal*是表现最好的系统，在88%（78%）的情况下，可以生成至少包含一个相关（高度相关）的存储库的成功建议，这是一个合理的高比例。在SuccessRate@4方面，*RepoPal*和*Combined*分别比*CLAN多出*12.82%和5.12%。就SuccessRate@5而言，这是一个更严格的标准，*RepoPal*和*Combined*以更高的幅度胜过*CLAN，即分别为*34.48%和24.14%。

[33]在计算精度的时候，有时也会计算召回率。请注意，我们没有计算召回率，因为我们不知道在我们的1000个资源库集合中相关和高度相关的资源库的总数。识别*所有*相关的和高度相关的资源库，需要太多的人工标签成本。

表二：成功率。*RepoPal* VS.*CLAN* VS.*合并的*

办法	成功率 (得分≥4)	成功率 (得分≥5)
回拨宝	88%	78%
CLAN	78%	58%
合并的	82%	72%

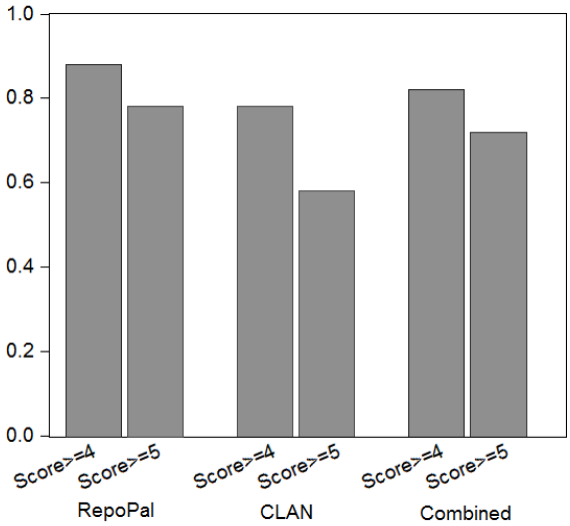


图4：成功率

表三：置信度。*RepoPal* VS.*CLAN* VS.*合并*

办法	样本量	中位数	平均值
回拨宝	250	4.0	3.52
CLAN	250	3.0	2.93
合并的	250	4.0	3.51

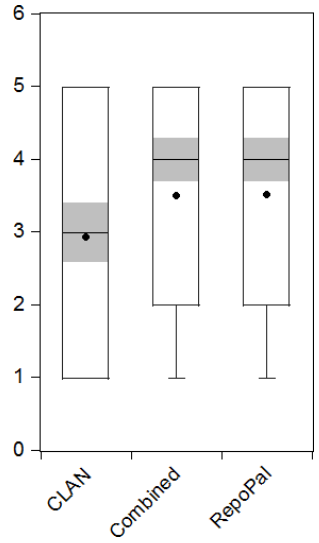


图5：置信度箱形图

我们注意到，*CLAN*产生的22%和42%的查询结果分别不包括被评为相关（4）或高度相关（5）的单一资源库。这表明只使用JDK API方法调用来描述存储库的局限性。许多JDK API方法调用是通用的，并不能完全描述存储库中实现的应用程序的语义。*RepoPal*的启发式方法能更有效地捕捉应用程序的语义，因此它能更好地识别类似的资源库。另外，将*CLAN*与*RepoPal*结合起来并不能提高性能，这突出表明包括JDK API方法调用可能会造成噪音，因为不相关的资源库可能会使用类似的方法调用（例如，java.util.ArrayList的方法被广泛的应用所使用）。

RepoPal在SuccessRate@4和SuccessRate@5方面分别比CLAN高出12.82%和34.48%。

B. RQ2: 信心

表三和图五显示了信心的实验结果。图5是一个方框图，显示了*RepoPal*、*CLAN*和*Combined*各自收到的250个评分的分布情况。根据表格和箱形图，*RepoPal*和*Combined*在250个参与者的评分中具有非常相似的平均和中位信心得分，这比*CLAN*的结果高。*RepoPal*和*Combined*在平均信心方面分别比*CLAN*高出20.14%和18.80%。*RepoPal*和*CLAN*的信心中值为4（相关），而*CLAN*的信心中值为3（中立）。

在50个查询中，*RepoPal*对35个查询的平均置信度高于*CLAN*，对两个查询的平均置信度相同。在32次查询中，*组合的*平均置信度比*CLAN高，在*6次查询中的平均置信度相同。

我们进行了Wilcoxon签名等级测试[13]来评估*RepoPal*对*CLAN*的改进是否在统计学上有意义，我们发现P值为0.032。因此，*RepoPal*对*CLAN*的改进在95%的置信度下是显著的。我们还进行了同样的测试来评估*Combined*比*CLAN*的改进是否有统计学意义，我们发现P值为0.0020，这表明*Combined*比*CLAN*的改进在95%的置信水平下是显著的。

RepoPal和Combined在平均信心方面分别比CLAN好20.14%和18.80%。这些改进在统计学上是显著的。

C. RQ3: 精度

表四显示了*RepoPal*、*CLAN*和*Combined*在50次查询中的中位数和平均精度。我们注意到，*RepoPal*和*Combined*的中位精度和平均精度都高于*CLAN*。*RepoPal*和*Combined*的中位精度为0.6，其平均精度分别为0.55和0.56。它们的平均精度分数分别比*CLAN*高出41.03%和43.59%。图6是显示50个查询中平均精度分布的框图。我们注意到，*CLAN*的上四分位数大大低于*RepoPal*和*Combined*的。

表四：精确度。*RepoPal* VS.*CLAN* VS.*综合*

办法	样本量	中位数	平均值
回拨宝	50	0.6	0.55
CLAN	50	0.4	0.39
合并的	50	0.6	0.56

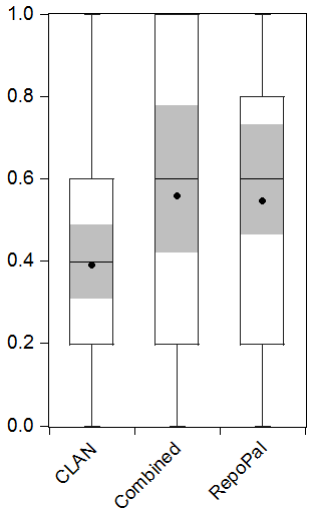


图6：精度箱形图

在50个查询中，*RepoPal*在33个查询中比*CLAN* *的*精度高，在6个查询中精度相同。在23次查询中，*Combined 的*精度高于*CLAN*，*在*21次查询中，*Combined 的精度与 CLAN 相同。*

再次进行Wilcoxon签名等级检验，以检验*RepoPal*和*Combined*相对于*CLAN* *的*改进在精度上是否具有统计学意义。与*CLAN*和*Combined*相比，*RepoPal*的P值是 *与CLAN*相比较，$4.007e^{-5}$ 和 $3.132e^{-7}$，分别为 这表明，*RepoPal*的改进和 *综合起来*比*CLAN*有明显的统计学意义。

*RepoPal*和*Combined*在平均精度方面分别比*CLAN*高出41.03%和43.59%。这些改进在统计学上是显著的。

VI. 对有效性的威胁

在这一节中，我们讨论了对我们的系统和实验的有效性的威胁。对有效性的威胁主要分为对内部有效性的威胁，对外部有效性的威胁，以及对构造有效性的威胁。我们还介绍了我们采取了哪些措施来减少这些威胁。

A. 对内部有效性的威胁

对内部有效性的威胁与实验偏见有关。我们强调在参与者和用于评估三个系统的资料库方面的两个威胁。

参与者。经验性的评价是基于50名参与者的得分。一些因素可能导致一些

对研究结果有效性的威胁；这些威胁包括：参与者对Java和GitHub的熟悉程度，参与者给予认真评价的动机，以及参与者对相关性标准的一致性。

尽管我们保证所有参与者都说自己熟悉Java和GitHub，但他们的熟练程度并没有被我们独立评估。对Java语言和GitHub知识的缺乏可能会影响参与者的判断。由于所有学生参与者都来自浙江大学计算机系，并学习了足够的技术课程，而且所有专业参与者都是知名技术公司的开发人员或测试人员（阿里巴巴有限公司[34]和网易有限公司[35][36]），因此这种威胁是有限的。

同时，如果参与者对评估资源库的相似性没有兴趣或动机，他们也可能做出不负责任的选择。我们通过选择那些对我们的研究感兴趣的参与者，并要求他们花足够的时间来理解资源库，从而将这种威胁降到最低。

参与者之间评价标准的不一致也可能产生负面影响。我们试图通过将一个查询的三个系统输出分配给同一个参与者来减少这种情况。因此，这个参与者在他/她的评级中的严格或宽松，将公平地分配给所有被评估的系统。由于每个查询只被分配给一个参与者，所以没有信息表明这些结果在不同的参与者之间有多可靠。如果我们有更多的参与者，我们可以把同一个查询分配给多个参与者，并计算出评分者之间的可靠性。请注意，过去许多需要用户研究的研究由于参与者数量有限，也没有计算出评判者之间的可靠性，例如[5]，[7]。

储存库。在这项研究中，我们只用50个查询来重新找到类似的存储库，以评估*RepoPal*、*CLAN*和*Combined*。我们也只从999个资源库（即1,000个减去作为查询的那个）中检索类似的资源库。在未来，我们计划使用更多的查询、存储库和参与者来减少对有效性的威胁。

储存库的质量也构成威胁。如果一个资源库的质量不高，可能没有描述或规范，编码风格也不好，参与者就很难给出正确的评价。我们选择了拥有超过20颗星的资源库来建立数据集。直观地说，这些受欢迎的资源库应该有更好的质量。类似的使用星级过滤资源库的策略在之前的许多研究中也做过，例如，[12]，[14]。虽然*CLAN*并不要求资源库必须有星星，但如果我们不限制星星的数量，*CLAN*很可能会检索到许多与查询资源库有类似API调用但质量不高的资源库，这也会损害*CLAN*的性能。

B. 对外部有效性的威胁

对外部有效性的威胁主要涉及到我们的研究和实验的可生成性。我们强调

[34] http://www.alibaba.com/
[35] https://en.wikipedia.org/wiki/NetEase [36] http://www.163.com/

在这项工作中考虑的编程语言和存储库的星级数量方面的威胁。

编程语言。GitHub上有许多用Java以外的语言（如Python、PHP、C++）编写的仓库，或多种编程语言的组合。RepoPal不是为单一语言设计的，可以应用于所有GitHub仓库。然而，由于我们想将*RepoPal*与*CLAN*进行比较，而*CLAN*只支持Java，所以我们在本研究中只关注Java仓库。我们计划在未来对*RepoPal*与其他用各种编程语言编写的仓库进行评估。

星的数量。当仓库的星级数减少时，RepoPal的检索质量就会下降。然而，正如之前所讨论的，大多数星级低的GitHub仓库也是低质量的，因此不利于被重用。

C. 对结构有效性的威胁

对建构有效性的威胁与我们的评价指标的适宜性有关。在这项工作中，我们使用了与McMillan等人[5]和Thung等人[7]的最密切相关的工作所使用的相同指标。这些指标是。SuccessRate@T, confidence 和 precision。这些指标是众所周知的指标，在以前的许多研究中也被使用[5]、[7]、[15]、[16]、[17]、[18]。

VII. 相关工作

寻找类似的存储库。与我们的方法最接近的工作是McMillan等人[5]和Thung等人[7]所做的研究。McMillan等人提出了一种方法*CLAN*，它使用API使用模式比较项目之间的相似性[5]。他们在超过8000个Java应用程序上评估了他们的技术，发现他们的方法比以前提出的技术有更高的精度。Thung等人提出了一种技术，根据SourceForge上与项目一起提到的软件标签来推荐类似的资源库[7]。他们进行了一项用户研究，结果表明他们的技术优于只使用Java API方法调用的JavaClan。

不幸的是，GitHub不支持仓库标签，McMillan等人的方法只依赖于API使用模式。在这项工作中，我们提出了一种新的方法，以解决之前的方法在识别GitHub上的类似仓库方面的局限性。它依赖于两个信息来源，即GitHub星级和自述文件，这在之前的工作中是没有使用的。我们还将我们的方法与McMillan等人在Java仓库上的工作（即*CLAN*）进行了比较，证明我们的工作优于他们的工作。我们没有将我们的方法与Thung等人的工作进行比较，因为他们的方法依赖于标签，而GitHub上的存储库没有标签。

软件推荐系统。关于软件推荐系统的研究已经有很多了。Bajracharya等人提出了一种结构性语义索引（SSI）技术，该技术根据API使用的相似性将单词与源代码实体联系起来，以推荐API的使用实例[19]。Thung等人提出了一种技术，推荐使用关联规则挖掘（基于当前库的使用情况）和协同过滤（找到其他类似项目使用的库）向开发者提供库[20]。他们对500个Java项目的评估显示了高召回率。Bauer等人提出了一种技术，通过利用基于标识符的概念定位和静态分析来检测源代码的重新实现[21]。Teyton等人提出了一种方法，分析软件项目中的源代码变化，这些项目从一个第三方库迁移到另一个，并提取旧库和新库之间的功能映射[22]。

我们的工作与上述研究是正交的：我们为一个给定的查询库推荐类似的库，这与上述工作解决的问题不同。

软件归类。有几种方法将项目分为不同的类别。Kawaguchi等人提出了一种技术MUDABlue，它使用源代码并应用潜在语义分析（LSA）来自动确定来自软件系统集合的不同类别，并将这些系统分类到上述类别中[6]。他们还实现了一个基于网络的界面来可视化不同的类别，并将他们的技术与之前提出的一些基于信息检索的技术进行比较。Wang等人提出了一种基于SVM的方法，通过聚合来自多个资源库的不同在线资料对软件项目进行分层分类[23]。他们对超过18000个项目进行了实验，发现他们的技术在精确度、召回率和F-measure方面有明显的改善。这些研究将项目分为不同的类别，然而，一个类别中可能有*许*多项目（平均有数百个[23]）。给定一个查询项目，不可能使用上述方法来区分同一类别的项目。

代码搜索引擎。一些研究提出了源代码搜索引擎，例如Exemplar [24], Sourcerer [25], SNIFF [26], Portfolio [27], SpotWeb [28], Parseweb [29], 和S6 [30]。这些搜索引擎恢复符合某种自然语言查询的源代码片段。在这项工作中，我们考虑一个不同但相关的问题，即在给定查询库的情况下，识别类似的库。

关于GitHub的研究。许多研究都分析了GitHub中的仓库（repositories）。例如，Bissyande等人研究了100,000个GitHub项目，以考察各种编程语言的普及性、互操作性和影响[31]。Ray等人分析了700多个项目，了解编程语言对软件质量的影响[32]。Vasilescu等人分析了数千个项目，并对GitHub用户进行了调查，以研究性别和任期多样性对团队生产力和营业额的关系[33]。与上述研究不同的是，我们关注的是一个正交的问题，即识别GitHub上的类似存储库。

VIII. 结论和未来工作

在GitHub上检测类似的存储库可以帮助软件工程师重用源代码，识别替代的实现，探索相关的项目，找到可以贡献的项目。

发现代码盗窃和剽窃，等等。之前有很多方法被提出来识别类似的应用，不幸的是，这些方法对GitHub来说并不理想。一种方法仅依赖于API方法调用的相似性[5]，而另一种方法则依赖于GitHub中不存在的标签[7]。他们没有利用两个可以直观地帮助识别相似仓库的信息来源，即GitHub的星星和自述文件。在这项工作中，我们提出了一个名为*RepoPal*的新技术，它利用了这两个信息来源。它基于三种启发式方法工作。首先，在很短的时间内被相同的人命名的仓库有可能是相似的。第二，由相似的用户命名的资源库可能是相似的。第三，那些自述文件内容相似的软件库很可能是相似的。在这项研究中，我们对*RepoPal*针对1000个库运行的50个查询进行了评估，并将其与*CLAN*进行了比较。我们的实验结果表明，*RepoPal*在成功率、置信度和精确度方面都能胜过*CLAN*。

在未来的工作中，我们计划通过在*RepoPal*的评估中加入额外的查询、存储库和参与者来减少对有效性的威胁。此外，我们还计划纳入更多的信息来源，以进一步提高*RepoPal*的有效性。

鸣谢

作者感谢所有参与本研究的开发者。本研究得到了中国科技部国家重点科技研发计划（编号：2015BAH17F01）的支持。

参考文献

[1] C.Liu, C. Chen, J. Han, and P. S. Yu, "GPLAG: Detection of software plagiarism by program dependence graph analysis," in *KDD*, pp.872-881, 2006.

[2] T.Sager, A. Bernstein, M. Pinzger, and C. Kiefer, "Detecting similar java classes using tree algorithms," in *MSR*, pp.65-71, 2006.

[3] C.McMillan, N. Hariri, D. Poshyvanyk, J. Cleland-Huang, and B.Mobasher, "推荐用于快速软件原型的源代码", in *ICSE*, pp.848-858, 2012.

[4] A.Mockus, "开放源码软件中的大规模代码重用", in *FLOSS*, 2007.

[5] C.McMillan, M. Grechanik, and D. Poshyvanyk, "Detecting similar software applications," in *ICSE*, pp.364-374, 2012.

[6] S.Kawaguchi, P. Garg, M. Matsushita, and K. Inoue, "Mudablue: an automatic categorization system for open source repositories," in *APSEC*, pp.184-193, 2004.

[7] F.Thung, D. Lo, and L. Jiang, "Detecting similar applications with collaborative tagging," in *ICSM*, pp.600-603, 2012.

[8] C.D. Manning, P. Raghavan, and H. Schu¨tze, *Introduction to Information Retrieval*.剑桥大学出版社，2008年。

[9] S.C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6。
p.391-407, 1990.

[10] G. Gousios, B. Vasilescu, A. Serebrenik, and A. Zaidman, "Lean ghtorrent:Github数据的需求，"*《第11届挖掘软件存储库工作会议论文集》*，第384-387页，ACM，2014。

[11] E.Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*, pp.92-101, ACM, 2014.

[12] B.Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pp.155-165, ACM, 2014.

[13] F.Wilcoxon，"通过排名方法进行个体比较"，*《生物统计学公报》*，第1卷，第6期，第80-83页，1945。

[14] C.Casalnuovo, P. T. Devanbu, A. Oliveira, V. Filkov, and B. Ray, "Assert use in github projects," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*, pp.755-766, 2015。

[15] S.Kawaguchi, P. K. Garg, M. Matsushita, and K. Inoue, "Mudablue:Mudablue: An automatic categorization system for open source repositories," *Journal of Systems and Software*, vol.79, no.7, pp.939-953, 2006.

[16] C.McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, "Portfolio: finding relevant functions and their usage," in *Software Engineering (ICSE), 2011 33rd International Conference on*, pp. 111-120, IEEE, 2011.

[17] M.Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C.Cumby, "A search engine for finding highly relevant applications," in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 1, pp.475-484, IEEE, 2010.

[18] C.McMillan, M. Grechanik, D. Poshyvanyk, C. Fu, and Q. Xie, "Exemplar:寻找高度相关应用的源代码搜索引擎"，*《软件工程》，IEEE Transactions on*，第38卷，No.5, pp. 1069-1087, 2012.

[19] S.K. Bajracharya, J. Ossher, and C. V. Lopes, "Leveraging usage similarity for effective retrieval of examples in code repositories," in *FSE*, pp.157-166, 2010.

[20] F.Thung, D. Lo, and J. Lawall, "Automated library recommendation," in *WCRE*, pp.182-191, 2013.

[21] V.Bauer, T. Volke, and E. Jurgens, "A novel approach to detect unintentional re-implementations," in *ICSME*, pp.491-495, 2014.

[22] C.Teyton, J.-R.Falleri, and X. Blanc, "自动发现相似库之间的函数映射", in *WCRE*, pp.192-201, 2013.

[23] T.Wang, H. Wang, G. Yin, C. Ling, X. Li, and P. Zou, "Mining software profile across multiple repositories for hierarchical categorization," in *ICSM*, pp.240-249, 2013.

[24] M.Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C.Cumby, "A search engine for finding highly relevant applications," in *ICSE*, pp.475-484, 2010.

[25] E.Linstead, S. Bajracharya, T. Ngo, P. Rigor, C. Lopes, and P. Baldi, "Sourcerer: mining and searching internet-scale software repositories," *Data Min. and Knowledge Discovery*, vol.18, no. 2, pp.300-336, 2009.

[26] S.Chatterjee, S. Juvekar, and K. Sen, "Sniff:一个使用自由格式查询的Java搜索引擎，"在*FASE*，第385-400页，2009年。

[27] C.McMillan, M. Grechanik, D. Poshyvanyk, Q. Xie, and C. Fu, "Portfolio:寻找相关的功能和它们的用法，"在*ICSE*，第111-120页，2011。

[28] S.Thummalapenta 和 T. Xie, "Spotweb:通过挖掘网络上的开放源代码来检测框架的热点和冷点"，在*ASE*，
pp.327-336, 2008.

[29] S.Thummalapenta和T. Xie, "Parseweb:一个在网络上重用开放源代码的程序员助手"，在*ASE*中，第204-213页，2007。

[30] S.P. Reiss, "基于语义的代码搜索", in *ICSE*, pp.243-253, 2009.

[31] T.F. Bissyande´, F. Thung, D. Lo, L. Jiang, and L. Re´veille`re, "Popularity, interoperability, and impact of programming languages in 100,000 open source projects," in *Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)*, pp.303-312, 2013.

[32] B.Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering（FSE）*, pp.155-165, 2014。

[33] B.Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik, P.Devanbu, and V. Filkov, "Github团队中的性别和任期多样性", in

Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI), pp.3789-3798, 2015.