数据收集和分析

GitHub 存储库和用户

弗拉基斯科·查齐阿西米迪斯

侏儒信息学 SA 21 安东尼斯·特里西斯海峡。希腊塞萨洛尼基 57001 + 30 - 2310804150

f.chatziasimidis@gmail.com

贷款

希腊塞萨洛尼基亚里士多德大学信息学系 54124 塞萨洛尼基+ 30 - 2310991910

stamelos @ CSD 认证组

摘要

在本文中，我们介绍了 GitHub 数据的收集和挖掘，旨在了解 GitHub 用户行为和项目成功因素。我们通过 GitHub API 收集了关于大约 lOOK 项目和 10K GitHub 用户/ /这些项目所有者的信息。随后，我们对这些数据进行统计分析，通过 k-means 算法离散化特征值，最后通过 weka 应用 apriori 算法找出关联规则。假设项目的成功可以通过下载的基数来衡量，我们只保留了下载基数高于 1000 次下载阈值的规则。这些结果为 GitHub 生态系统提供了有趣的见解，并为 GitHub 项目提供了七条成功规则。

关键词

GitHub，开源，GitHub API，关联规则，k-means，离散化，项目成功

介绍

开源软件(asS)现在是程序员和计算机用户群体中非常流行的一种软件，这种趋势在未来几年中似乎会加剧。到目前为止，OSS 社区已经产生了许多成功的项目，如 Linux、Apache 服务器、Mozilla Firefox、LibreOffice 等。因此，开放源码软件生态系统(包括开放源码软件项目和社区)在过去几年里一直在积极研究。其中重要的一点是理解 OSS 范式是如何运作的，以及为什么运作良好。本文分析了 GitHub 门户的 OSS 项目，并挖掘了这些项目的成功规则。

GitHub 是一家 OSS 锻造公司，成立于 2008 年，旨在简化代码共享。GitHub 逐月快速增长，现在可能是 OSS 项目最大的存储库。2008 年，GitHub 对 41157 名用户和 38423 个项目进行了评分，2012 年，这些数字增加到 2763000 名用户和 4614306 个项目。Scuh 的数字显示了 GitHub 的增长率，并表明开放源码软件用户非常喜欢 GitHub。gitHub 如此成功的一个主要原因是项目托管基于流行的 Git 系统。

相关著作

在开放源码软件数据收集和分析领域已经进行了几项研究。例如，Gousios 等人。在[1]描述了使用 GitHub rest API 收集数据的详细过程以及所收集数据的确切模式。OSS 数据挖掘领域的另一个有趣的研究是查瓦拉、阿鲁纳萨拉姆和戴维斯·[2]。在本研究中，作者从 Sourcforge 收集数据，然后使用关联规则网络(ARN)来发现项目特征之间的关系。在另一项相关研究中，高、黄和玛蒂

检索整个 Sourceforge 数据库，使用非负矩阵分解(NMF)方法选择独立的重要特征，并应用数据挖掘技术(聚类和汇总)找到规则和相关特征[3]。Raja 和 Tretter 提出了另一项研究，是 Logistic 回归(LR)、决策树(DT)和神经网络的结合，旨在找出解释 OSS[4 成功的因素。我们最近和最相关的研究是伊曼纽尔、瓦尔多约、伊斯蒂扬托和穆斯托法·[的研究。在这篇论文中，作者从 Sourceforge 收集了 OSS 项目的数据集，然后尝试用数据挖掘 3 项集关联规则来发现成功规则。以前 Bibi、Stamelos 和 Angelis 将关联规则(AR)和分类回归树(CART)结合起来，以确定项目属性和开发[项目所需的努力之间的逻辑关联。此外，Stamelos 和 Bibi 使用封闭源项目数据集上的关联规则来估算基于类比的项目成本[7]。与以前的作品相比，本文的贡献有两方面: (1)我们试图发现结合项目特征和用户特征的规则；(2)我们在 GitHub 相对新的蓬勃发展的生态系统上提供和讨论这些结果。

收集的数据

为了收集 GitHub 数据，我们使用了 GitHub 休息 API。此 API 旨在返回关于用户的所有信息，包括存储库及其特性。因此，我们创建了一个 java 应用程序，它使用给定的用户名，并将所有与该用户相关的信息存储在 PostgreSQL 数据库中。我们没有使用[1]中描述的工具，因为它以事件驱动的方式检索数据，忽略不活动的项目和不活动的用户。在我们的工作中，我们需要随机的用户和项目，以便全面了解 GitHub 生态系统。

出于实际性能原因，我们不得不对 GitHub 用户群进行随机抽样。我们使用了谷歌大查询服务，通过它我们检索了与 GitHub 相关的数据。从 2008 年到今天，我们选择了与一些用户合作，这些用户看起来一直很活跃。对于所有这些用途，我们进行了 15 天的收集过程。数据收集过程中的主要性能问题是 GitHub API 每小时有 4000 个请求的限制，包括身份验证请求。这个问题降低了收集速率，迫使我们的收集器应用程序停止，直到请求限制在一段时间间隔后被取消。

对于每个用户，我们检索的信息是姓名、电子邮件，

博客、创建日期、uri、位置、公共存储库的数量、私有存储库的数量、公共注册表的数量、数量

私人注册，布尔变量 isHirable，合作者数量，追随者数量，追随者数量和

htWs:/ /开发者。谷歌。com /大查询!

用户上次操作的日期。Gist 指的是完成特定操作的有限数量的代码。功能 isHirable 表示用户是否有意被雇用。

对于我们检索到的每一个存储库，

描述、创建日期、分叉数量、主页 url、名称，

所有者、git url、未解决问题的数量、boolean 是 fork (表示存储库是否是另一存储库的 fork)、boolean 有下载、下载次数、boolean 有问题、boolean 有 wiki、boolean 是私有的以及挂钩的数量。术语 fork 表示一个存储库，它是另一个存储库的扩展。这两个存储库被认为是不同的。术语钩子表示软件项目功能的改变。

并非所有 GitHub API 信息都有助于定量分析。例如，用户名、电子邮件或网址不能为数据分析提供任何有用的信息。因此，我们选择为每个项目保留 13 个特征，用于后续分析。存储的功能包括:存储库主语言、分叉数量、未解决问题数量、wiki、挂钩数量、存储库年限(从创建日期提取)、存储库用户的国家、

存储库用户的公共存储库数量(参考存储库用户)、存储库用户协作者数量、存储库用户追随者数量以及存储库用户追随者数量。

数据收集过程

收集过程是通过一个定制的 java 应用程序来实现的，该应用程序将一个用户名列表作为输入，并将所有 GitHub 用户信息存储在数据库中。收集过程的伪代码是:

```
开始_时间_分钟+ -获取当前时间戳大小_聚焦器_列表+- 10153 sum _ oCrequests +- 0，同时(真) \ 0
而(尺寸_聚焦器_列表> O 和总和_ oCrequests<4000 ) >
获取与当前用途相关的信息存储请求+的信息到数据库总和，同时( -；iz _ ocREPO _ list > O 和
sum_oCrequests<4000 ) >
ˇ
将用户的信息存储到数据库中
Sum _ Jequests++
ˇ
waite 60- (当前时间分钟开始时间分钟) )开始时间分钟+ -获取当前时间戳}
```

这个 java 应用程序运行了 15 天，从 2013 年 4 月 1 日到 2013 年 4 月 15 日，产生了 192732 个存储库和 10153 个用户的数据集。

数据分析

我们的目标是挖掘收集的数据并产生描述有趣数据关系的关联规则。我们选择使用 apriori 算法，可通过 weka 工具获得。Apriori 算法[9]是一种从大型数据库中提取关联规则的简单算法。该算法通过识别数据库中频繁出现的单个项目，并将它们扩展到更大的项目集，只要这些项目集在数据库中足够频繁地出现。Apriori 确定的频繁项目集可用于确定关联规则。

Apriori 算法适用于离散值，而 GitHub 的大多数特征是以连续值给出的。为了解决这个问题，我们选择使用 k-means 算法来离散特征值。如[8]所述，离散连续值的一种常见的无监督方法是运行 k 均值算法，并发现值范围内的中心点。接下来，可以将每个值放入具有最近中心的聚类中。K-means 是一种迭代算法，它以簇的数量作为输入，迭代计算新的中心，直到它以中心的稳定排列收敛。这样，连续值被中心的离散值所取代。

如上所述，数据收集产生了两个 csv 文件，一个用于用户，一个用于存储库。这两个文件都被导入 spss 并进行分析。首先，计算用户和存储库数据的描述性统计数据，揭示数据集的主要特征。接下来，我们将 k-means 算法应用于被选择参与关联规则提取过程的 13 个特征，并且我们为每个特征找到了五个中心(离散值)。之后，我们重新计算每个特征的值，并创建了包含离散化的 13 个特征的 arff 文件。最终运行 apriori 算法并产生关联规则。Apriori 算法要求用户设置支持和置信度阈值。我们提醒读者，支持指的是规则左侧项目集的频率，置信度指的是同一规则左侧项目集所在行集合中规则右侧的频率。

我们数据集的一个重要特征是下载次数的分布。正如 OSS 生态系统中经常观察到的，很小比例的项目下载量非常大，很大比例的项目下载量很少甚至为零。换句话说，这个特性的值遵循 zipf 分布。例如，通过查看描述性统计数据，我们发现下载次数超过 1000 次的项目比例为 246 / 192735 = 0.001，因此如果我们将支持阈值设置为 0.8，我们将永远不会找到包含超过 1000 次的项目下载的规则，因为该项目的支持度仅为 0.001。为了解决这个问题，我们在不支持过滤但置信度为 0.75 的情况下运行该算法，并保留检测稀有项目集的规则，但是我们保证，如果在规则的左边找到某些项目集，右边肯定会包含> 1000 的

项目下载。通过这种方式，我们成功地研究了 GitHub 子集中最受欢迎的(就下载而言)以及最成功的 OSS 项目。

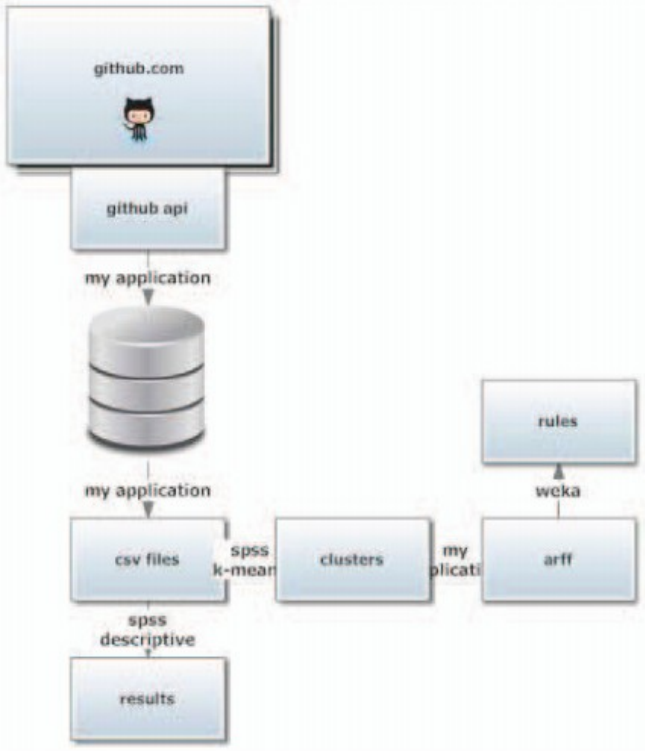


图 9。收集和分析过程

结果

图一至图八和表一提供了描述数据集的信息。在大多数情况下观察到 zipf 分布。图一显示了下载的 zipf 分布。很明显，大量项目只有一次下载，很小一部分项目有大量下载。图表和统计表都是从至少有一次下载的项目子集中提取的(2642 / 192735)。在图 2 中，我们可以看到项目中未决问题的分布。从至少有一个未决问题的项目子集提取的图表和统计表数据(2642 / 192735)。在图 3 中，展示了每个项目的分叉分布。从至少有一个分叉的项目子集提取的图表和统计表数据(3lO88/192735)。在图 4 中，我们可以看到项目中语言的分布。图 5 显示了用户的百分比及其公共存储库的数量。正如我们看到的，大多数用户拥有从 l0 到 40 个存储库。在图 6 中，我们可以看到大多数用户有 2 - 5 名合作者，因此我们可以安全地说，大多数项目都有小型开发团队。最后，图 7 和图 8 显示了追随者的数量和追随者的数量。一小部分用户拥有大量的追随者和追随者，但大多数人拥有少量甚至一个追随者或追随者。

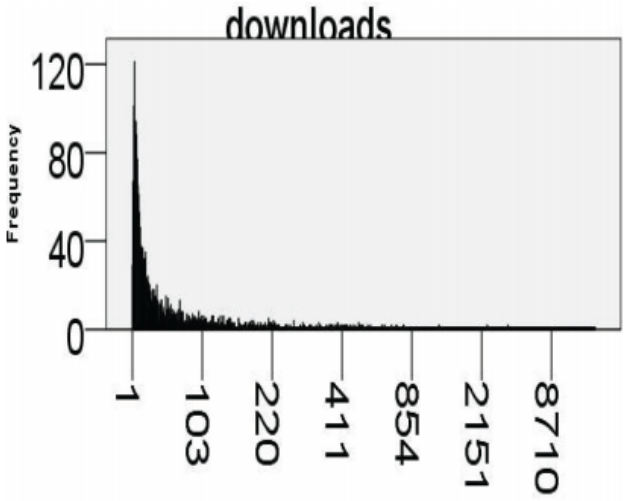


图 1 .每个项目的下载量

描述统计学				
	中间	变化	最低限度	最高的
下载	5025	149948160 42，07	1	5548233
未决问题	5，34	312，308	1	1233
叉	5，94	967，81	1	2204
公众的 仓库	19，13	2163，16	0	3582
合作者	2，39	39，83	0	347
被跟踪者	14，74	17788，20	0	12789
追随者	25，65	22438，51	0	6846

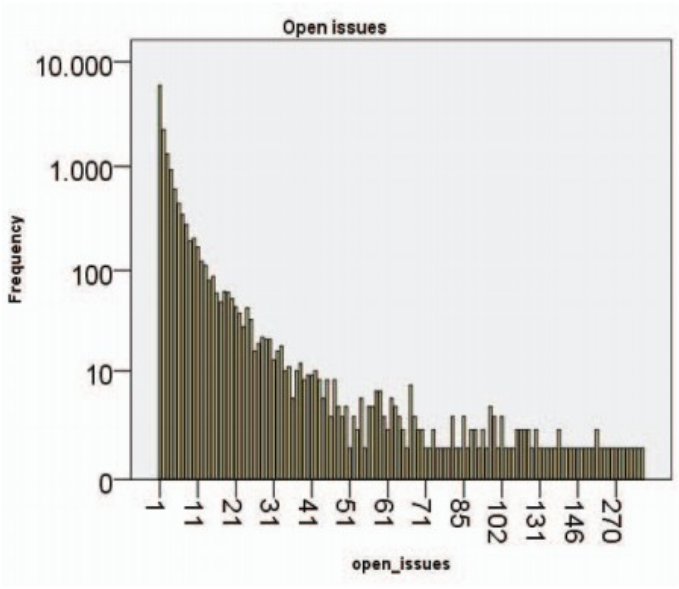


图 2.0 未决问题

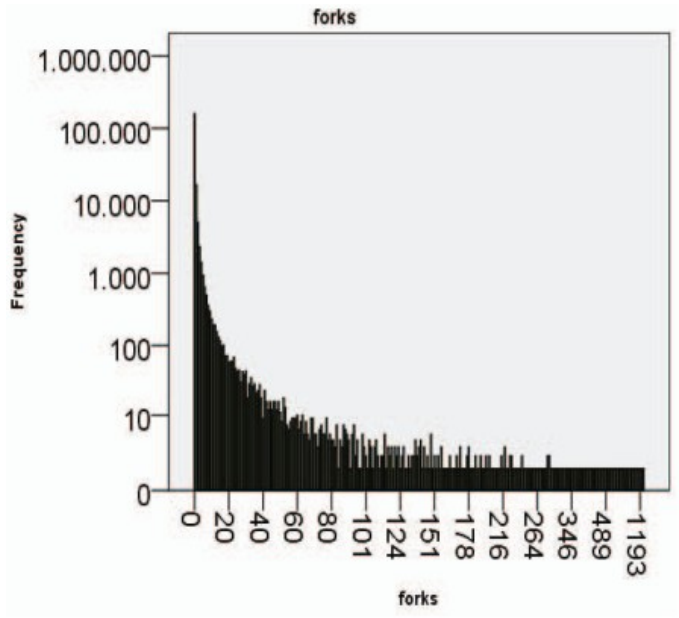


图 3 .叉子

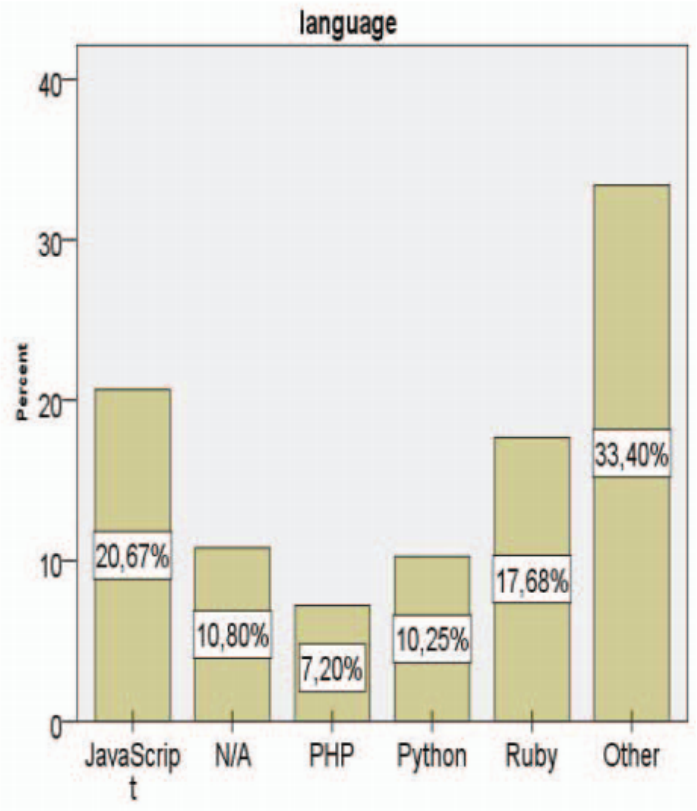
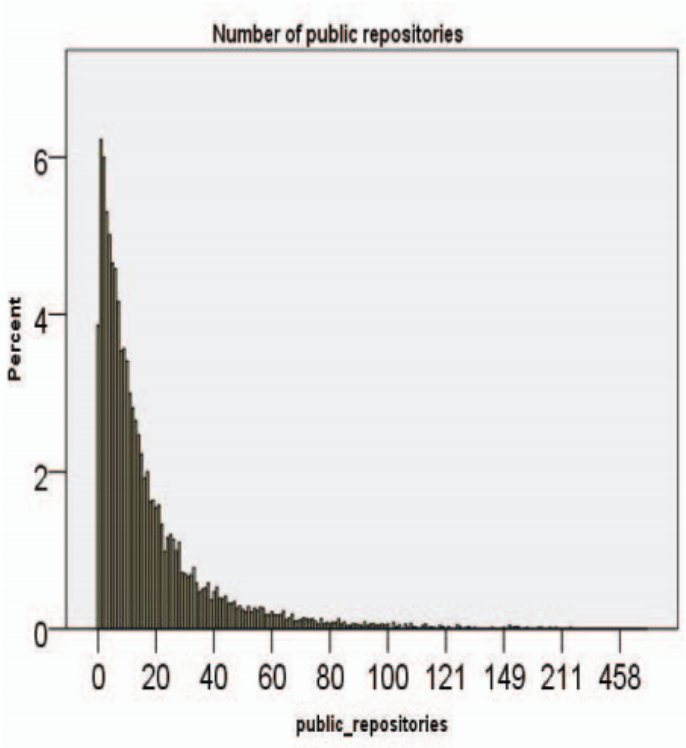


图 4。存储库语言



数字。公共储存库数量

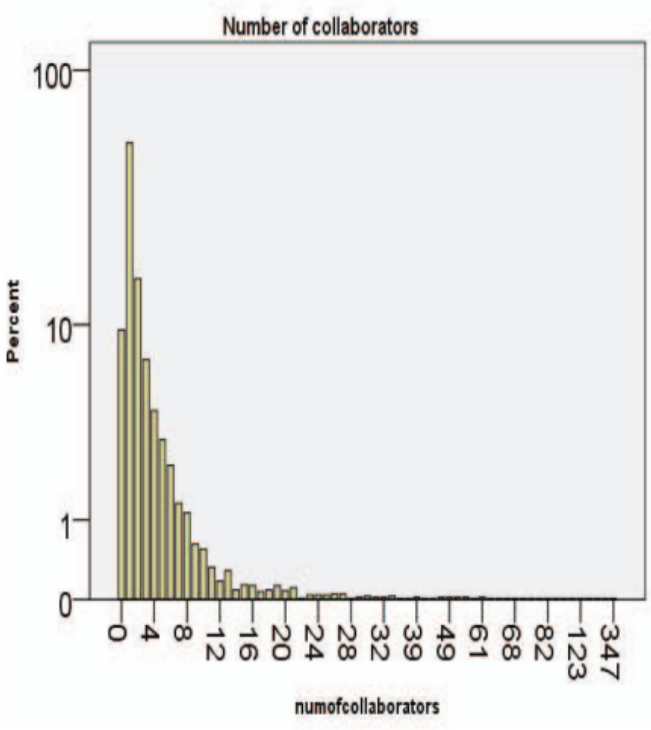


图 6。合作者人数

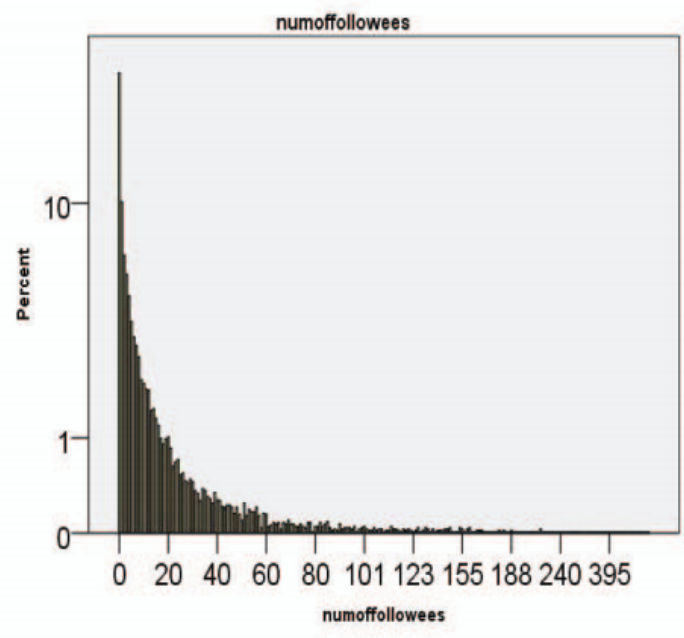


图 7。被跟踪人数

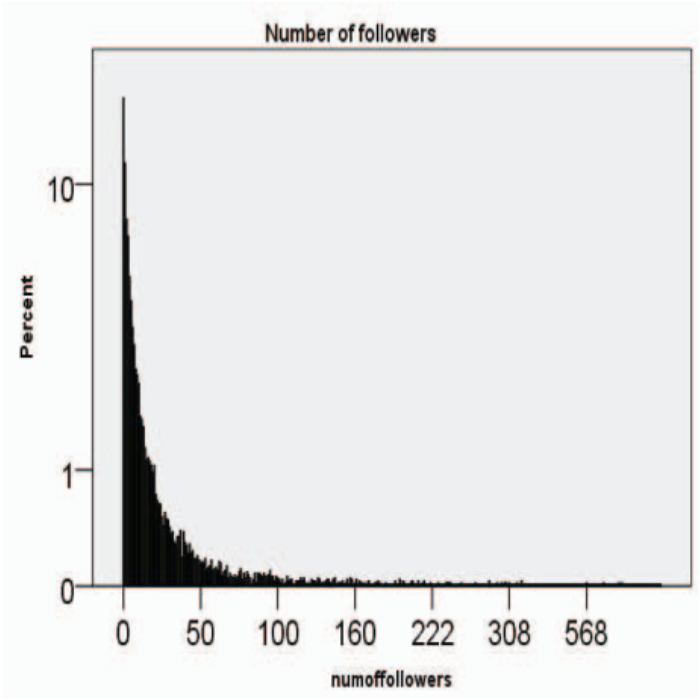


图 8。追随者人数

成功规则

在本节中，我们简单列出了检测到的最成功的关联规则。

1 .openlssues =接近_ 26 用户 userLastAction =接近_ 4 numOfFollowees =接近_ 8 >

下载量= 1000，置信度 0.99

2 .repoAge =旧_ 36 用户 userLastAction =近_ 4 个用户协作者=近_ 2 个用户跟踪者=近_ 8 = >下载量= 1000，置信度 0.81

3 .Repage = midle _ age _ 24 UserLastAction =近 4 User PublicRepositories =近 12 numOfFollowers =近 13 numOfFollowees =近 8 = >下载量= 1000，置信度为 0.78

4 .Repage = midle _ age _ 24 UserLastAction =近 4U SerPublicStores =近 12 NumFollowers =近 13 numOfFollowees =近 8 = = >

下载量= 1000，置信度 0.78

5 .hasWiki =真实报告=旧_ 36 numOfFollowees =近_ 8 = = >下载= 1000，更有信心 0.77

6 .报告=旧_ 36 isHirable =假 6142 = = >下载= 1000，可信度为 0.77

讨论

按照我们的方法，我们最终发现了 6 条成功规则。第一条规则告诉我们，如果项目有接近 26 个未决问题，并且用户在过去 4 个月里在 GitHub 环境中采取了一些行动，并且用户有接近 8 个追随者，那么项目下载

量超过 1000 次，概率为 99 %。换句话说，这条规则说，如果一个项目是活动的，它的所有者是活动的，并且跟随少数其他用户，它很有可能成功。在这一点上，我们可以观察到，成功的项目似乎有一些既不是零也不是很大的未决问题。直觉上，零未决问题可能揭示了一个不活跃的项目，而大量未决问题意味着一个不活跃的项目或低质量的项目。

第二条规则告诉我们，如果项目的年龄接近 36 个月，这可能被认为不是一个新项目，并且项目用户在过去 4 个月中采取了行动，合作者的数量接近 2 个，并且紧跟在 8 个其他用户之后，那么这个项目的下载量超过 1000 次，概率为 81 %。换句话说，除了规则 1 考虑的特性之外，规则 2 还检查项目的成熟度和开发团队的规模。正如我们看到的，开发团队相对较小的成熟项目似乎更成功。这个发现值得进一步研究，以了解为什么有四五个合作者的项目看起来不那么成功。

第三条规则规定，如果项目年龄接近 24 个月，项目所有人在过去四个月里采取了一些行动，项目所有人有 12 个公共存储库，项目所有人有 13 个追随者，8 个追随者，那么项目下载量超过 1000 次，概率为 78 %。此规则将项目的成功与项目的成熟度、用户活动、存储库的用户数量和用户社交活动相关联。在此规则中，新项目用户公共存储库出现在 12 点附近，这在以前的规则中是没有发现的。此项目显示，如果用户有少量公共存储库，这些存储库更有可能成功。这听起来很合理，因为与拥有大量公共存储库的用户相比，拥有少量公共存储库的用户可能会更加投入。这条规则的另一个值得注意的因素是，成功项目的用户似乎不够社交化。

第四个规则是对第三个规则的扩展，因此它在左侧具有与规则 3 相同的项目集，加上开发团队的规模。有了这条规则，我们可以看到，和以前一样，小团队成功运行项目的概率更高。

第五条规则是，如果项目有 wiki，已经成熟，并且它的用户跟随少数其他用户，那么它有超过 1000 次下载，概率为 77 %。这条规则中的一个新项目是 Wiki，这是开源项目的一个非常重要的特性，因为它帮助其他用户下载和使用它，使项目更受欢迎。这条规则表明，如果一个项目

成熟，它的所有者不愿意被雇佣，那么它很有可能会成功。这个规则也是有道理的，因为如果一个开发者已经有了一个下载量超过 1000 的项目，那么他可能对另一个项目不感兴趣。

结论和今后的工作

本研究试图发现将开源项目和所有者特征与项目成功相关联的模式。我们检索了大约 10 年的数据.从非常受欢迎的 GitHub 存储库中查找项目和 10K 用户，并对其进行预处理后进行分析，以实现价值离散化。我们已经能够为成功的项目产生六条关联规则，使用下载次数作为项目成功的代表。作为第一个结论，通过组合所有六个规则，成功的 GitHub 项目似乎表现出以下特征。

成熟

高活性

对其他用户的支持

主动所有者

拥有少量其他项目的业主

小型开发团队

拥有少量追随者的所有者

拥有少量追随者的业主

没有被雇佣欲望的所有者

目前我们的研究中存在一些有效性威胁。一个项目的成功可以用不同的方法来衡量，例如考虑到它被翻译成的语言数量。此外，我们通过观察数据分布，将项目分为成功和不成功的 1000 个项目，作为下载的阈值。在这一点上，对于未来的工作来说，寻找更精细的方法来选择这个阈值是一个好主意。研究中可以改进的另一点是，与 k 均值算法相比，我们通过寻找更好的方法来离散遵循 zipf 分布的连续值，从而离散数据集特征的值。用于提取规则的其他挖掘算法可能会给出更好的结果。最后，GitHub API 数据本身可能存在一些问题，例如项目活动的级别和强度不可区分，因为用户的一次简单提交决定了最后一次行动日期的值。

我们计划下载完整的 GitHub 数据集并重复我们的分析，解决上述研究问题。我们目前的数据可以在 http://sweng.csd.auth.gr/githubData.zip 获得，并且可以被希望分析 GitHub 这一特定子集的研究人员用于未来的研究。

参考

[1]乔治·古西奥斯，MSR '13。该数据集和

工具套件(2013)

[2] Sanjay Chawla，Bavani Arunasalam 和 Joseph Davis.2003 .挖掘开源软件(OSS)数据

使用关联规则网络，PAKDD'03 第七届太平洋-亚洲知识发现和数据挖掘进展会议记录，第 461 - 466 页

2003 年

[3]丁，秦永·高，黄英平，格雷格·玛迪，数据

开源软件社区中的挖掘项目历史，N AACSOS 会议 2004，宾夕法尼亚州匹兹堡，2004

[4]乌兹玛·拉贾和玛丽埃塔·特里特，2006 年。调查

开源项目成功:一种数据挖掘方法

模型制定、验证和测试，SAS 用户组国际会议，论文 070 - 31，SUGI 2006

[5] Andi Waju Rahardjo Emanuel，Retantyo Wardoyo，Jazi Eko Istiyanto，Khabib Mustofa。2010 年。开放源码软件的成功规则

使用数据挖掘 3 项集关联规则的项目，

DCSI 国际计算机科学杂志，2010 年第 7 (6)期

[6]斯塔马蒂亚·比比，伊万尼斯·斯塔梅洛斯，左厄里斯·安吉利斯. 2008。

结合概率模型进行解释

生产力估计，信息和软件技术，50 (7 - 8)，第 656 - 669 页，2008 年

[7]伊奥尼斯·斯塔米罗斯·斯塔玛蒂娅·比比。基于类比的成本估算配置与规则，第八届基于知识的软件工程联席会议记录，第 317 - 326 页，2008 年

[8]詹姆斯·多尔蒂，罗恩·科哈维，Mehran Sahami，斯坦福大学计算机科学系，连续特征的监督和非监督离散化，Armand Prieditis & Stuart Russell 编辑。机器学习:第十二届国际会议记录，1995 年

[9]勒凯什·阿格拉瓦尔和罗摩克里希南·斯里坎特。1994 年。快

在大型数据库中挖掘关联规则的算法。

1994 年第 20 届 VLDB 会议记录