GitHub 用户数据分析

刘天扬 许定胜

一. 引言

GitHub 作为近年来快速发展的开源项目托管平台和开发者社交网络，已有上千万的注册用户以及开源项目，其中包含大量与开发者相关的信息。为了充分挖掘 GitHub 数据中蕴藏的价值并加以利用，帮助用户在搜索时更容易找到自己感兴趣的领域中的高质量的用户与项目，本次实验通过分析 GitHub 数据集中的各种交互数据，衡量开发者的影响力并对开发者进行分类，找到与目标用户具有较高关联度的高价值用户及开源项目。

二. 问题描述

1.本次实验基于 GitHub 用户数据和日志数据，使用多种算法对数据进行处理计算和分析。对于用户的评价指标应该与该用户创建的仓库质量高低以及该用户的社交网络有关。GitHub 数据集中 ods 层的用户数据表中数据容易处理但是可用信息少；日志数据表中虽然蕴藏大量的用户行为数据，但是数据较为杂乱。如何从这些数据集中提取出实验所需要的信息？

2.对数据分析后使用单一指标对用户的质量进行评价太过片面。如何综合数据对用户得到一个较为客观全面的评价？

三. 方法

在 GitHub 中，一个高质量的用户应该具备以下几个素质：在社交网络中具备一定的影响力；在社交网络中具备一定的活跃度；拥有一定数量的高质量代码仓库。根据数据的特征，本次实验将使用 PageRank，HITS，H-因子，用户所拥有的仓库属性等 6 项指标来对开发者进行比较，并最终根据这 6 个指标对开发者进行综合评估，得到对开发者较为客观的评价。考虑到日志数据集十分庞大，在现有的设备条件下使用至今为止的所有日志数据进行分析研究不太现实，所以本次实验只使用 2021 年一年的日志数据。以下先对数据集的特征进行介绍，再逐一介绍评价指标与算法。

1. 数据预处理

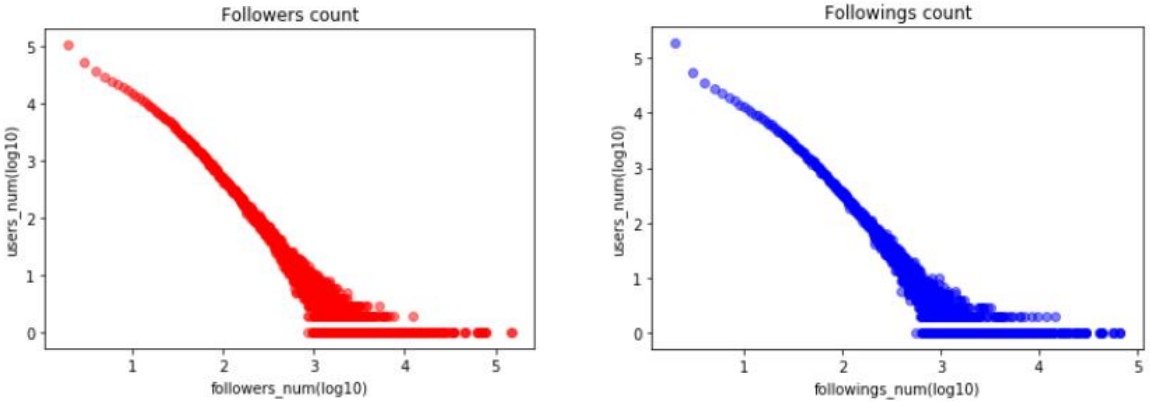
后续实验中对用户的评价指标有相当一部分需要依赖用户所拥有的仓库进行操作，而在 ods 层和 dim 层的数据中，并没有将用户与其拥有的仓库一一对应的数据集。所以在开

始实验之前，需要先对日志数据进行操作，筛选出用户与其拥有的仓库以及仓库的属性，并将它们对应起来方便后续操作。

日志数据中记录了 GitHub 上各种类型的操作的数据，为了筛选出用户与用户拥有的仓库，首先将事件类型设为 CreateEvent 对日志数据进行爬取，对得到的数据去重后获得用户与用户拥有仓库一一对应的数据。接下来需要收集各个仓库的具体属性。这里以仓库发生最后一次操作时的数据为准，对下载的数据进行去重，去重后将之前处理好的“用户-仓库”数据表与仓库属性数据表做 join，这样就得到了每个用户和该用户拥有的仓库以及仓库对应属性的数据。完成以上预处理操作之后，接下来对开始对数据的分析。

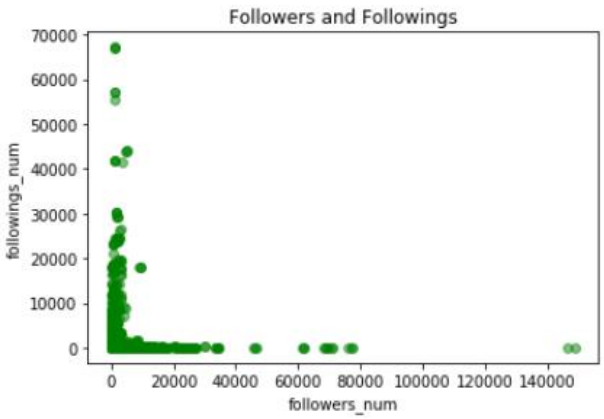
2. Follow 网络中的用户特征分析：

本次实验统计 ods_github_users 中用户的 follower 和 following 数据并绘制散点图，为了更直观地展示特征分布，图中横轴定义为某一特征取值加 1 的对数（取 log10），纵轴为各个特征取值对应的用户数量加 1（取 log10）。



由这两张图可以看出，follower 和 following 数量级在 10^3 以下的用户在 GitHub 中占了绝大多数。

接下来对用户的 following 和用户 follower 进行统计，绘制散点分布图如下所示。横轴代表用户 follower 数量，纵轴表示用户 following 数量。



从图中可以看出，大部分用户集中分布在靠近零点的区域，在这个区域中，用户的following 数和 follower 数都相对较小。这一数据说明，与用户直接相连，即直接受该用户影响的用户是十分有限的。从图中还可以观察到：在开始的时候，随着用户follower 数量的增加，用户 following 数量也增加，呈正相关，但是持续的范围很小。之后随着用户 follower 数量增加，关注的用户数量逐渐减少了。另外一个比较极端的现象就是当用户 follower 数量很大时，following 的数却相对很小；following 数很大时follower 数量反而很小。例如有“Linux 之父”之称的 Linus Torvalds 在 GitHub 拥有超过 158k 的 follower，但是他的 following 却是 0。这一现象在各种社交网络中都十分常见，他们往往是各个领域中的知名人物，他们的 follower 数量有相当一部分是基于名人效应。在 GitHub 这样汇聚了大量开发者的社交网络中，用户的 follower 数量除了与用户在现实中的社交关系有密不可分的关系之外，一个用户如果拥有很大数量的follower，在一定程度上也代表这个用户受到其他开发者认可。因此，将基于 follow 网络的 PageRank 和 HITS 作为衡量高质量用户的指标是合理的。、

3. 评价指标及算法：

3.1. PageRank：

PageRank 是一种链接分析算法，它是一种衡量网页重要程度的方式。Page Rank 算法通过计算网络页面的链接数量和质量来确定网站的重要性。该算法基于以下假设：

如果一个网页被很多其他网页所指向，那么说明这个网页比较重要；

如果一个网页被重要网页所指向，那么说明这个网页比较重要。

计算公式如下：

$$rank(p_i) = \frac{1-q}{N} + q\sum_{\mathrm{p_j}} \frac{rank(p_j)}{D(p_j)}$$

其中，rank(p_j)，是网页p_j的 PageRank 值，N 为所有网页的总数，D(p_j)是页面p_j的链出网页数目。q 为阻尼因子，通常设为 0.85，类比用户按照网页链接进行浏览的概率， 1-q 则是指用户随机跳转到一个新页面的概率。

PageRank 算法步骤：

输入：用户 Follow 图 G=<V,E>

输出：用户节点 PageRank 值

1. 对所有节点，初始化节点的 PageRank 值为 1；
2. 基于图遍历每一个节点，利用公式更新该节点的 PageRannk 值；
3. 重复步骤 2，直到算法收敛；
4. 将用户的 PageRank 值降序排列，返回结果。

本次实验将 follow 网络中的每一个用户视为一个网页，通过计算出用户的 PageRank 值来得出在网络中较为重要的用户，并以此作为评价指标之一。

3.2. HITS：

HITS 是一种链接分析算法。在 HITS 算法中，每个页面被赋予两个属性：Hub 值和 Authority 值。根据页面两个属性值的大小，网页可以被分为 Hub 页面和 Authority 页面。页面的 Hub 值等于所有该页面指向的页面的 Authority 值的和；页面的 Authority 值等于所有指向该页面的 Hub 值的和。该算法基于以下假设：

一个高质量的 Authority 页面会被很多高质量的 Hub 页面所指向；

一个高质量的 Hub 页面会指向很多高质量的 Authority 页面；

计算公式如下

$$a(u) = \sum_{\mathrm{e}(v,u)\in E} h(v)$$

$$h(u) = \sum_{\mathrm{e}(u,w)\in E} a(w)$$

$$a(u) = \frac{a(u)}{|max(a(u))|}$$

$$h(u) = \frac{h(u)}{|max(h(u))|}$$

HITS 算法步骤：

输入：用户 Follow 图 G=<V,E>

输出：用户权威值列表，用户枢纽值列表

1. 对任意一用户 u，初始化 a(u)=1 和 h(u)=1；

2. 遍历图 G 中的每一个结点，对任意一个结点 u，遍历其所有粉丝结点,该点的 a 值为所有粉丝结点的 h 值的和；

3. 遍历图 G 中的每一个结点，对任意一个结点，遍历其所有粉丝结点，该点的 h 值为其粉丝结点的 a 值的和；

4. 对得到的 h 值列表和 a 值列表分别进行归一化；

5. 重复步骤 2,3,4 直到前后两次计算得到的结果差值小于预定义的阈值（无限小）

6. 将所有用户的 a 和 h 值按降序排列，返回结果。

本次实验将 follow 网络中的每一个用户视为一个网页，通过计算出用户的 H 值和 A 值来得出在网络中较为重要的用户，并以此作为评价指标之一。

3.3. H-因子：

H-因子，又称 H 指数，是一种用来评价学术成就的方法。一位学者的 H-因子为 h，表明该学者至多有 h 篇论文均被引用了至少 h 次。H 因子能够比较准确地反映一个人的学术成就，它不仅衡量了学者的发文数量也衡量了学者的发文质量。一个人的 H-因子越高，表明其论文影响力越大。

本次实验将每个用户视为学者，用户仓库视为该学者发表的文章，引入 H-因子来衡量用户的影响力，并以此作为评价指标之一。

3.4. 用户仓库属性：

在 GitHub 中，一名用户的仓库质量高低和影响力大小基本上决定了该用户在 GitHub 社交网络中的影响力。而仓库在 GitHub 中的影响力可以从多方面来评估，比如该仓库的 star 数量，fork 数等等。

本次实验筛选统计出 GitHub 中用户名下的所有仓库进行统计分析，主要通过 star，fork，pr 三个指标对用户进行评价，并以此作为评价指标之一。

3.5. 波达计数法：

波达计数法（Borda Count）是一种排序投票法。波达计数法通过投票上的排序来为候选项赋予积分，然后按照积分降序来对所有候选项最终排序。由于波达计数法的简洁有效性和易于计算，在群体决策、项目论证、人工经济评价、质量评估等方面有着广泛应用。其计算方法如下：

$$\mathrm{B(u)} = \sum_{\mathrm{i}\in\mathrm{I}} r_u(i)$$

其中，$r_u(i)$是对象 u 基于评估标准 i 的得分值，I 是评估标准的集合， B(u)是对象 u 的最终积分值，也就是最终用来排序的标准。

本次实验使用波达计数法对以上指标进行综合评价，并对用户进行打分，得到最终的用户质量分析结果。

四． 评价

1. PageRank 和 HITS 的有效性分析

1.1. 斯皮尔曼相关系数：

斯皮尔曼相关系数是用来衡量两个排序之间相关度的一种方式，计算公式如下：

$$\rho = 1 - \frac{6\sum(x_i - y_i)^2}{\mathrm{N}^3 - \mathrm{N}}$$

上式中，对一个具有 N 个对象的集合，x_i和y_i分为为对象 i 在两种度量得到的排序 X 和 Y 中的排序位数。 ρ 为相关系数值，属于范围 -1 到 1，如果两个排序一致，即完全正相关， p=1，如果两个排序完全负相关，那么 p-=1。

对用户的 PageRank 值和 HITS 算法中得到的 Authority 值进行排序，得到的结果计算斯皮尔曼相关系数，得到结果为 0.689，由此可以看出 PageRank 和 HITS 是强相关的。这一点从两个算法的节点更新方式和使用的数据中也可以部分看出。另外，基于 Follow 网络得到的 PageRank 值与直接通过用户 follower 数量排序得到的结果应该是强相关的，但是这不代表 PageRank 算法在这里使用是无意义的。因为在前文对 PageRank 算法计算方式介绍的时候提到，PageRank 的计算基于两个基本假设，两个假设可以被概括为数量假设和质量假设，即用户在 PageRank 中的排名不光与用户的 follower 数量有关，follower 的质量也被考虑在内。因此，我们可以认为，使用 PageRank 作为用户质量的评价指标之一，要明显优于直接对用户的粉丝数量进行排名得到的数据。HITS 算法在此处同理。

2. 案例分析：

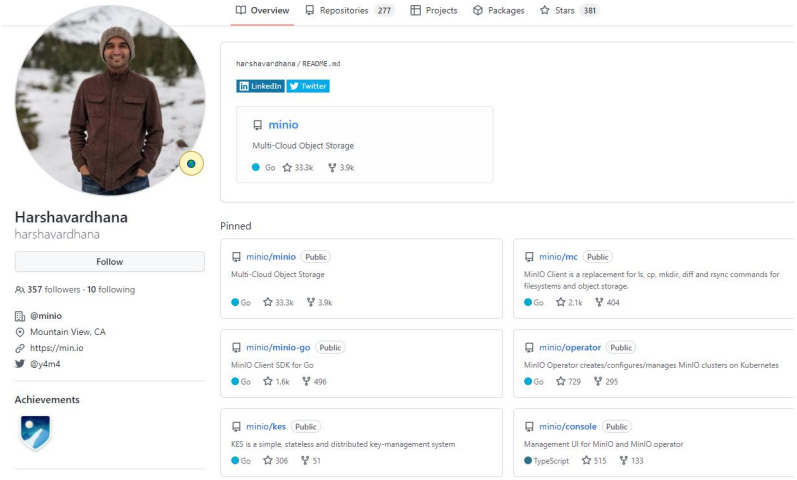
对所有评价指标分别进行排名，再由波达计数法计算得出用户的最终得分，得到结果如下：

pr	fork	star	H_index	PageRank	h	a	score
10810283	6422482	6422482	170270	1024025	14808551	1024025	622699
7571158	43972606	170270	6422482	905434	4624113	499550	17034772
11215289	42785357	15801806	483853	72895	35243344	810438	790842
52211928	53313357	63889819	1341245	150330	6391776	905434	319282
69343704	1341245	1884376	534619	343936	13808299	170270	399535
19560074	583231	46919888	47313	8030945	3076393	39191	493969
7911893	46058173	19358186	44816363	11601040	74522790	66577	455023
42283162	5118881	4591597	17034	230541	1308397	80	273120
37169284	349621	51722130	52195	98681	47697490	150330	2136245
60107403	9666939	6822714	9011267	8798027	82409010	1500684	1478487

这里需要指出，当仅使用仓库属性作为排名依据时，排名靠前的用户基本上被机器人包揽了；仅使用 PageRank 和 HITS 作为排名依据时，排名靠前的基本上都是各行各业的顶尖大佬。然而在本次实验计算最终得分之前，先将所有在当前评价指标中存在空值的用户进行去除，最终得到的结果中，机器人几乎不存在了。这是因为 GitHub 上的机器人在仓库操作上往往会远超真人，然而这样的机器人几乎没有社交关系，这样通过 PageRank 和 HITS 指标的空值就能去除这些机器人。同样的，具有名人效应但是在选取的数据的时间

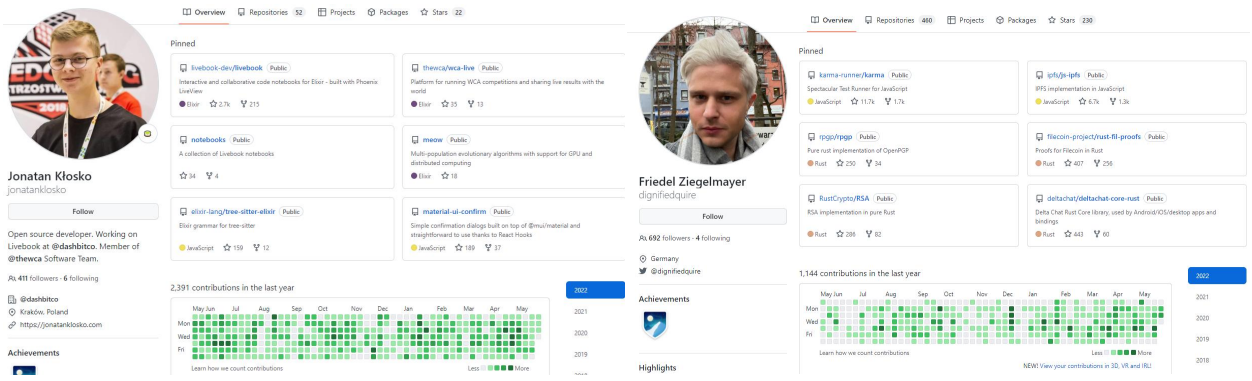
范围内没有过仓库操作的不活跃用户在这里也会被去掉。最终能够进入排名的用户需要同时具备社交网络中的高影响力和高活跃度等多种素质。

选择最终得分前三的用户查看其 GitHub 主页，排名第一的用户名为 Harshavardhana。



他曾经是 GlusterFS 的开发人员，现在是 MinIO 的创始人之一。该用户在过去一年里的活跃度相当高，他所创建的仓库也收获了大量的 star 和 fork。显然这是一个在 GitHub 社交网络中的高质量用户，然而用户本人的知名度并不高，在国内的网页搜索该用户，能够得到的相关信息少之又少。

另外两名用户分别是 Jonatan Klosko 和 Friedel ZiegeImayer。



这两位用户也有类似的属性。他们都在 GitHub 网络中拥有一定数量的 follower，而且在选取的数据的时间范围内有着很高的活跃度，他们所创建的仓库也得到了广泛的认可，符合我们对于 GitHub 这样的开发者社交网络中高质量用户的定义。

五. 相关工作

1.数据集的下载和处理

完整的GitHub数据集过大，全部下载并进行处理和使用并不现实，所以本次实验选择爬取2021年一年的数据进行操作。对于分月份爬取的数据，将全年的数据一次性读取操作也不现实，所以在正式实验之前，先将数据分季度读取到notebook中进行预处理操作后，再将处理好后的数据整合到一起方便后续的实验操作。

2.对最终得到的高质量用户进行了了解。

3.尝试使用louvain算法对所有用户进行社区发现，但是无法解决图数据过大算法运行时间太长得不出结果的问题。最终仅使用了100条和500条数据验证算法的有效性。

六. 结论

本次实验通过定义六个指标衡量用户质量，并最终由波达计数法综合给出最终评价来寻找GitHub中存在的高质量用户。通常我们寻找这样的开发者网络中的高质量用户，除了通过名人效应和自己现实的社交网络来判断之外，很难再有别的途径来找到一个高质量的用户。然而，名人效应意味着隐藏在社交网络中的大量的名气并不算大的优质用户难以为人所知，用户自己在现实中的社交网络又太小，由此能够找到的高质量用户十分有限。本次实验最终为我们所找到的用户从各个角度综合评价都完全可以用优秀来形容，通过本次实验提出的方法，可以有效帮助用户找到更多平时被他们所忽略的优质用户；对于那些不太为人所知但是有着良好的开发行为的用户，本次实验提出的方法能够让他们的工作被更多人发现。

参考文献

[1] 高明,胡卉芪,等. 随机游走及其应用. 数据科学与工程算法基础:120-128.

[2] Chris Ding,Xiaofeng He,Parry Husbands,Hongyuan Zha,and Horst Simon. PageRank,HITS and a Unified Framework for Link Analysis. Proceedings of the 2003 SIAM International Conference on Data Mining(SDM)

[3] Bornmann, L., Daniel, HD. Does the h-index for ranking of scientists really work?. Scientometrics 65, 391–392 (2005).
[4] Clint D. Kelly, Michael D. Jennions, The h index and career assessment by numbers, Trends in Ecology & Evolution, Volume 21, Issue 4, 2006, Pages 167-170
[5] 王姗姗. GitHub 用户数据分析与研究[D].大连理工大学,2018.

附录：分工

刘天扬：相关工作调研，文献查阅，数据处理，代码实现，结果分析，报告编写。

许定胜：相关工作调研，文献查阅，报告编写。

附录：进度

第一阶段（第 4-6 周）：确定选题，相关工作调研，确定方案

第二阶段（第 7-8 周）：数据预处理

第三阶段（第 9-10 周）：PageRank，HITS 代码实现

第四阶段（第 11-13 周）：完整代码实现

第五阶段（第 14 周）：分析结果，编写报告