# 1.介绍

Github是一个面向开源以及私有软件项目的托管平台。目前注册用户已经超过千万，托管的版本库数量已将超过两千万，其中不乏知名的开源项目，如：Ruby on Rails、Hibernate、phpBB、jQuery、Prototype、Homebrew等。GitHub于2008年4月10日正式发布[[8]](http://www.worldhello.net/gotgithub/01-explore-github/010-what-is-github.html" \l "id19)，相比始于1999年的SourceForge[[9]](http://www.worldhello.net/gotgithub/01-explore-github/010-what-is-github.html" \l "id20)和2005年的GoogleCode[[10]](http://www.worldhello.net/gotgithub/01-explore-github/010-what-is-github.html" \l "id21)，GitHub后来者居上。以2011年的数据从代码提交数量上看，GitHub已经超越其前辈[[11]](http://www.worldhello.net/gotgithub/01-explore-github/010-what-is-github.html" \l "id22)，如图1-1所示。

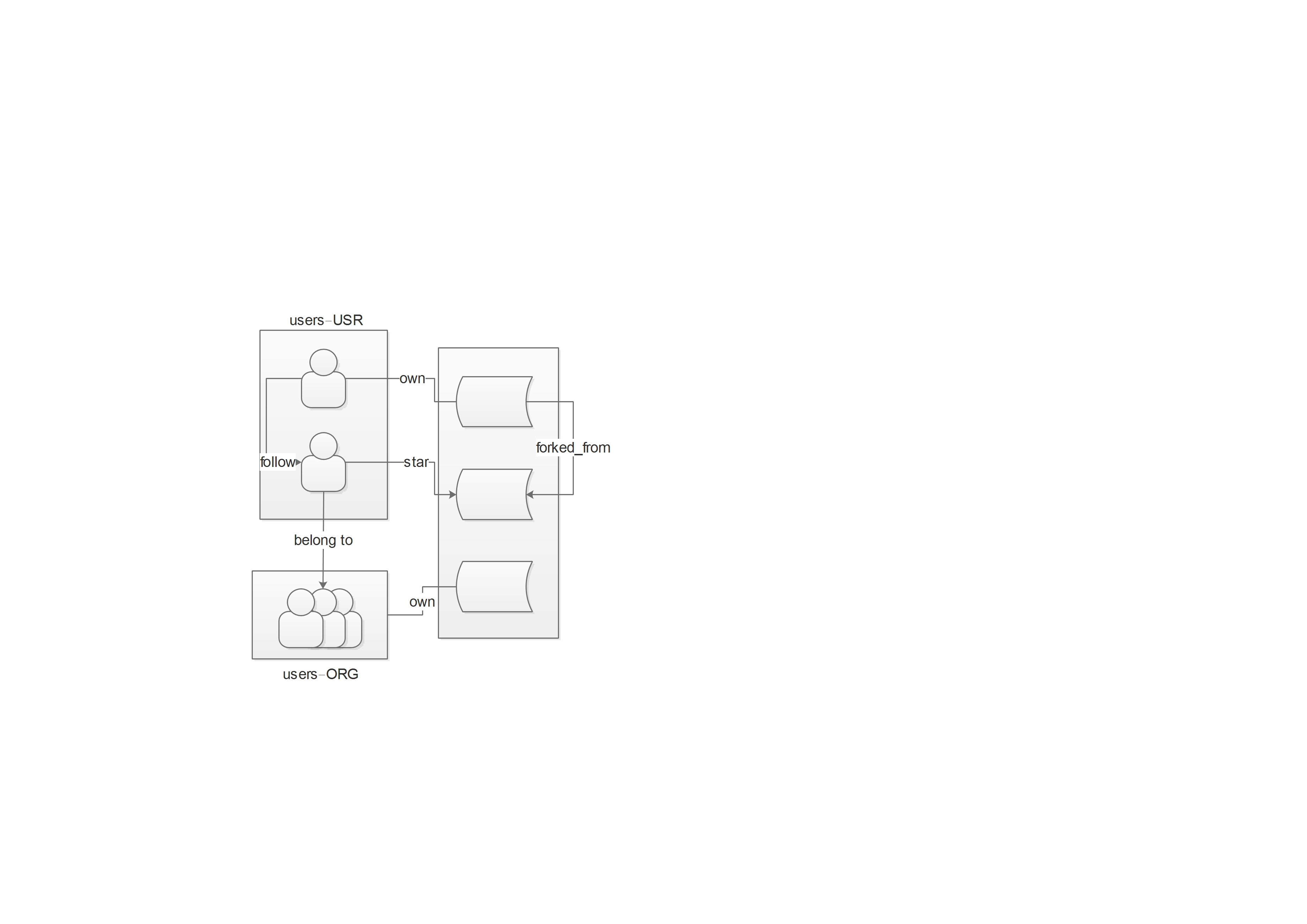
本文贡献：

1. 基于大量的数据，对用户特征进行了分析和提取
2. 多角度分析

# 2.相关工作

# 3. 本文方法

（相关方法介绍）

In this section, we will introduce our user influence ranking model. We evaluate user influence from five kinds of information (follow, watch, fork, activity including create projects, commit, open issues, open pull request). The motivation can be described as follows:

1. 高影响力的用户通常会被其他很多高影响力用户关注；
2. 一个好的枢纽用户会关注很多高权威的用户；一个好的权威用户会被很多好枢纽用户关注；
3. 高影响力用户的项目通常会受到更多关注；
4. 高影响力用户通常会有较多项目受到较多关注；
5. 高影响力用户的项目通常会被更多人fork；
6. 高影响力用户通常会有更多的项目被较多的fork
7. 较活跃的用户在未来可能会产生更多的影响

在Github中用户分为普通用户USR和组织用户ORG，在这里我们将其看成两种节点，那么Github网络就包含了三种节点，即USR, ORG和项目。节点之间的关系包括USR之间Follow关系，用户与项目之间own关系，star关系，USR与ORG之间归属关系，以及项目与项目之间fork关系。

如图1为整个网络的一个示例图。为了更清楚地理解整个Github网络，我们可以将其分成几个子部分，来从不同角度对用户影响力进行分别衡量，包括follow关系，star关系，fork关系并结合own关系，以及图中没有展现的用户活跃性。

**Follow：**follow关系中，仅有USR一种节点和follow关系一种边，结合基本假设一和基本假设二，我们使用经典的PageRank算法和HITS算法（将在下一节介绍）来对用户进行排序。

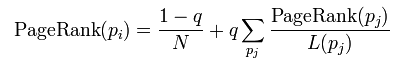
**Star：**在star关系图中，基于基本假设三，我们统计用户所有项目被star的总数，并据此对用户进行排序；基于基本假设四，我们引入H-index算法对用户进行排序，具体细节将在下一节介绍。

**Fork：**fork关系存在于项目与项目之间，我们结合用户与项目的拥有关系，将其转化为用户fork项目的关系，这样该关系与star关系类似，所以我们基于基本假设五和六，首先采用与star关系相同的方法进行分析；然后对用户fork之后的行为情况进行了分析。

**Activity：**在github中，一个用户可以创建项目、commit、pull request、open issue等。基于基本假设七，用户越活跃说明其越具有潜在的影响力。More active participation means the user’s higher level of preference and contribution. On the other hand, frequent activities bring more opportunities of interaction between users, which are important premise of influence.（用户通过更多的活动，一方面带来更多与他们交互的机会，另一方面也会产生更多贡献。）

## 3.1 PageRank算法

PageRank算法核心思想为，质量高的网页所指向的网页质量也一定高。其计算公式如下：

IMG_256是被研究的页面，IMG_257是IMG_258链入页面的数量，IMG_259是IMG_260链出页面的数量，而N是所有页面的数量。q为阻尼系数，一般取值为0.85，其意为用户到达某页面后并继续向后浏览的概率。

## 3.2 HITS算法

HITS算法是链接分析中非常重要且基础的算法，它涉及两个基本定义：Hub页面（枢纽页面）和Authority页面（权威页面）。权威页面即高质量的网页；枢纽页面即包含了很多指向高质量页面链接的网页。HITS算法基于如下两个假设：

假设1：一个好的“Authority”页面会被很多好的“Hub”页面指向；

假设2：一个好的“Hub”页面会指向很多好的“Authority”页面；

利用这两个基本假设，以及相互增强关系等原则，通过迭代计算，完成对网络中节点的度量。计算公式如下：



## 3.3 H-index算法

h-index ，又称为h指数或h因子（h-factor），是一种评价学术成就的方法。h代表“高引用次数”（high citations），一名科研人员的h指数是指他至多有h篇论文分别被引用了至少h次。h指数能够比较准确地反映一个人的学术成就。一个人的h指数越高，则表明他的论文影响力越大。

我们基于H-index方法和基本假设，提出H-watch方法，来衡量一个用户的在Github网络中项目成果。H代表“高watch次数”，一个用户的H-watch是指他至多有h个项目被watch了至少h次。H指数能够较准确的反应一个用户的项目成果，其H指数越高，说明其项目影响力越大，则用户影响力越大。

# 4. 数据分析

数据的预处理和数据分析

实验中我们使用的数据集是来自Github API[[1]](#footnote-0)的截止到2017年01月19日的数据。对于每个用户和项目，Github都提供了其相关信息，如图1所示，（a）为用户信息，（b）为项目信息。数据集中用户总数将近800万，如此大量的用户，其中必定有一些用户其影响力极小甚至可以说不具有影响力。这些用户的存在不仅可能会对我们影响力的度量产生干扰，并且会占用很大的时间和空间开销，因此，我们首先对数据集进行预处理。

1.1 数据预处理

我们首先对通过如下规则对用户和项目进行初步筛选。对用户：

1. 去除deleted标签为1的用户；

2. 去除fake标签为1的用户；

3. 去除没有follower的用户；

4. 去除没有project的用户；

对项目：

1. 去除边deleted标签为1的项目；
2. 去除不属于所选用户的项目；

筛选后的用户为我们实验中要衡量影响力的用户。预处理前后，数据集详细信息如下表1所示：

表1 数据筛选前后信息

|  |  |  |
| --- | --- | --- |
|  | 原始数据量 | 预处理后数据量 |
| users | 16188060 | 1884071 |
| projects | 46772903 | 20798840 |

1.2.1 Github 关系综述

通过对以上关系进行分析，我们得出几个可能与用户影响力相关的基础指标，用户粉丝数（userFollowersNum）、用户关注数（userFolloweesNum）、用户项目数（userReposNum）、用户项目被watch的数量（userWatchersNum）、用户项目被Fork的数量（userForkedNum）。为了了解这些指标分布，我们计算了他们的最大值、平均值、中位数和标准差离差率，结果如下表所示：

表2. 用户基本指标统计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FeatureName | 最大值 | 平均值 | 中位数 | 标准离差率(%) |
| userFollowersNum | 45112 | 6.89 | 733 | 1167.42 |
| userFolloweesNum | 196098 | 6.89 | 533 | 3919.87 |
| userReposNum | 24178 | 11.34 | 434.5 | 370.92 |
| userOriginalReposNum | 18090 | 6.34 | 216.5 | 339.75 |
| userForkReposNum | 23020 | 5.00 | 382.5 | 682.50 |
| userWatchersSumNum | 369085 | 20.08 | 2148.5 | 2764.62 |
| userForkedSumNum | 33342 | 3.40 | 657.5 | 2306.24 |

从表中我们可以得出，以用户粉丝数为例，平均每个用户有6到7个粉丝，中位数733，标准离差率较大，说明粉丝数分布两极化较严重。其他数据也呈现相似的规律，说明用户各个数据的分布是较分散的，大部分用户的实际数值与平均数值之间还是有差异的。

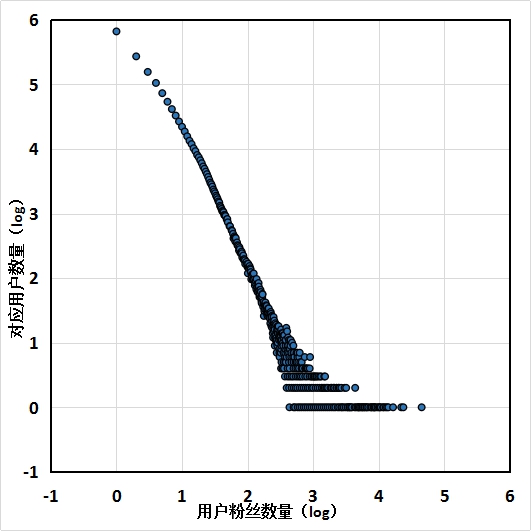
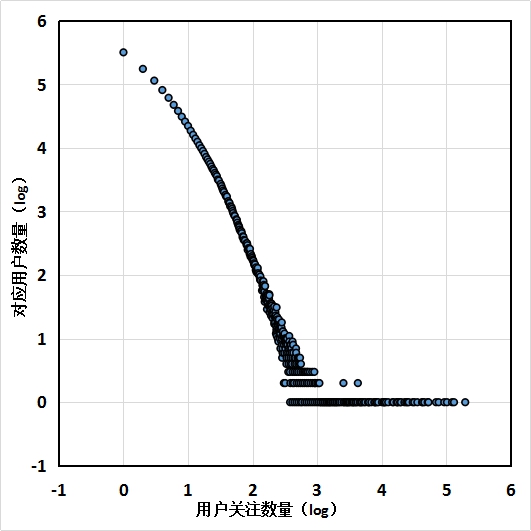
为了验证以上猜想，我们绘制以上指标的分布图。

1. 用户特征与用户数量分布图

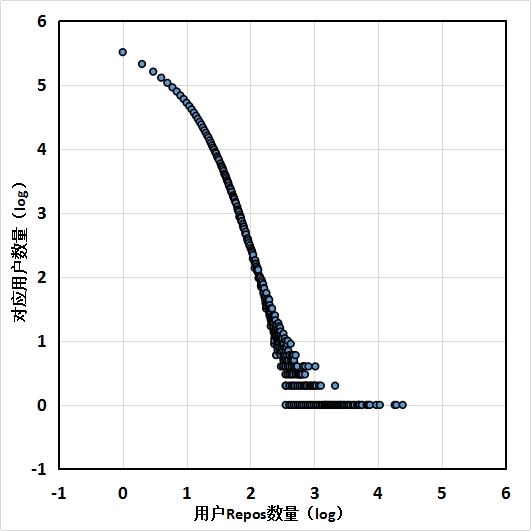
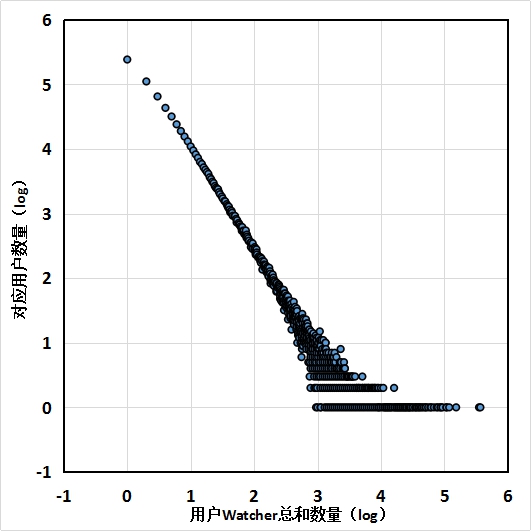
横轴表示用户粉丝数量，纵轴表示对应拥有某粉丝数量的用户数量，两者都取

log，通过以上各个子图，我们可以把握各个度量指标的总体分布情况。以图(a)用户粉丝数为例，最左上角的一个点表示有接近10的六次方，即100万用户没有粉丝。

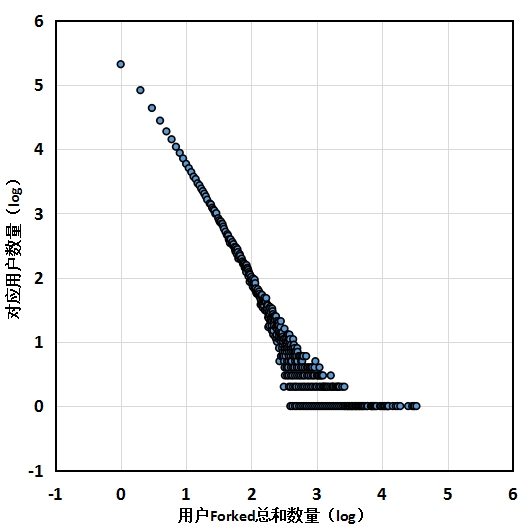
观察对比5个子图，我们还可以发现，相比较而言图(b)，(d)具有往外凸的趋势，而恰好其代表的用户关注数量和用户拥有项目数量是用户自己决定的，而不同于其他三者由网络或者说网络中的其他用户决定。从这也可以看出用户在网络中的主动性，主动关注其他用户以及主动创建项目。

(a) (b)

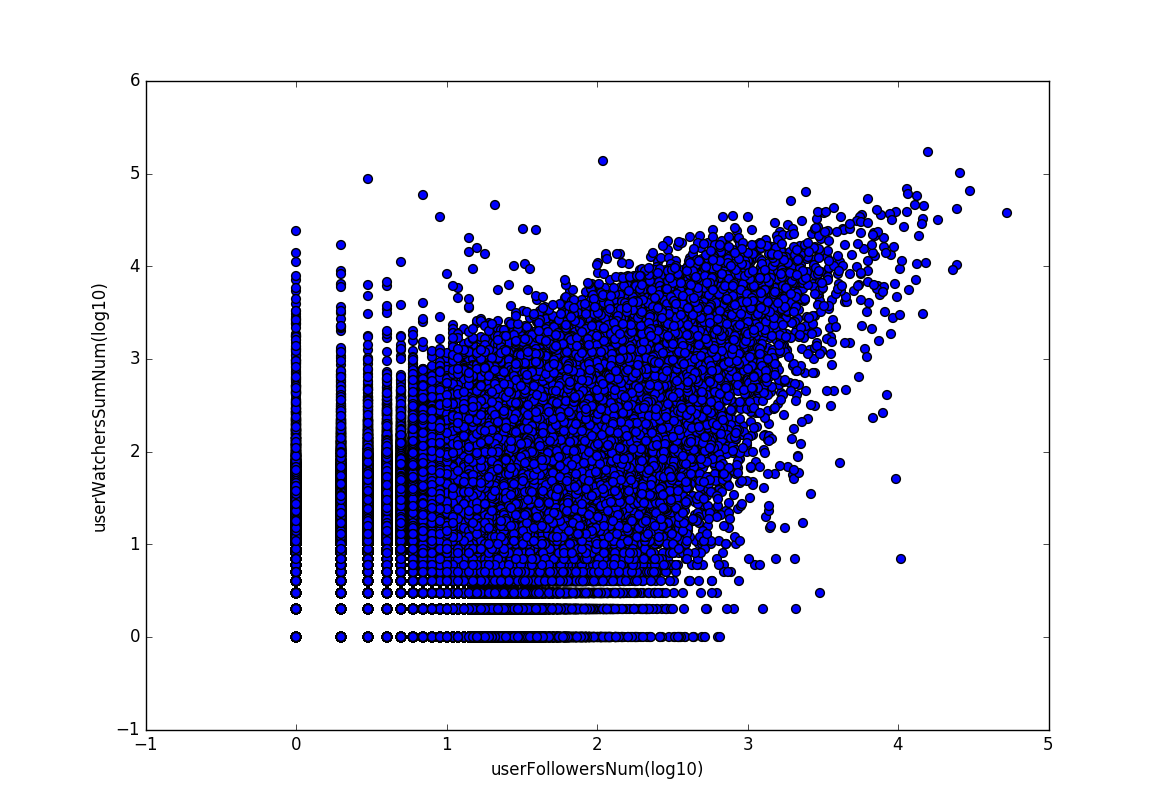


(c) (d)



(e)

1. 用户粉丝数量与用户数量关系图



用户具有较高影响力直观体现为用户粉丝数和用户项目的watcher数。因此我们对用户的Followers数量和用户项目的watcher数量之间关系进行了分析，如上图所示。很明显可以看出两者基本呈现正相关关系。但是存在两种特殊的情况：

1. 拥有较大的Follower数量，但是相对应的watchers数量较小；这种情况，通常是由于“名人效应”引起的，即某用户在实际中拥有比较受关注，所以加入Github后即收到较多的Follower。如RubyOnRails的创始人David Heinemeier Hansson，截止到2017.01.19他有10425个粉丝，但是其拥有的项目总共只收到了7次watch。
2. 拥有较多的watcher，但是Follower比较少。这种情况主要出现在组织中。这种组织通常有自己的产品项目，收到很多watcher。但是由于Github自2013年左右之后不再有Follow组织选项，所以其Follower数量统计是截止到2013年左右。

# 实验结果和分析

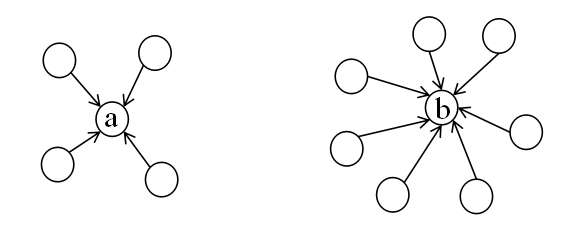
## 5.1 Follower

将社交网络引入项目托管平台是Github的创举。用户可以关注项目、关注其他用户进而了解项目和开发者动态。根据Github用户之间的关注和被关注的关系，我们构建Gihub社交网络，得到有向图G=<V, E>，V为节点，即用户集合；E为边的集合，用户A关注用户B，则存在一条A指向B的边。我们需要对用户影响力的分析，即量化每一个用户的影响力，然后找出影响力较大的用户。针对这个问题，我们基于网络G，提出以下几种算法。

### 5.1.1 用户粉丝数量

**算法思想**

衡量用户影响力算法中，最直接的方法就是统计分析用户的粉丝数量，即网络中节点的入度。因为我们分析的是用户的影响力，即用户对其他用户行为活动等影响度度量，直观上拥有更多粉丝的用户，更能够将信息传给他人。如图2所示，网络中节点a和节点b拥有不同的入度，其影响力也不同，从该网络中可以直接得出，用户b比用户a拥有更大的影响力。

图2 举例说明用户a和用户b在网络中结构

因此我们基于这样的假设：“受更多人关注的用户，越能够影响其他用户”，提出基于粉丝数量的用户影响力分析算法，即Github社交网络G中，入度越大，其影响力越大。

基于以上思想，**算法步骤**如下：

**算法 1：**

输入：Github社交网络有向关系图G=<V, E>；

输出：用户影响力度量以及排序

算法步骤：

1. 初始化列表L，L中存放对用户影响力的度量，初始值均为0；
2. 遍历关系图G中的每一个节点u，计算其入度In(u)，并将该值存入列表L，作为用户影响力的度量评分；
3. 对列表L中按照度量评分值降序排列；
4. 返回列表L；

**结果：**

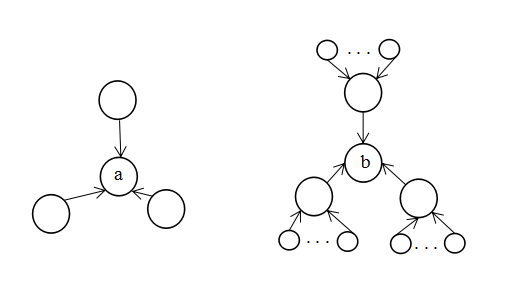
|  |  |  |  |
| --- | --- | --- | --- |
| **rank** | **login** | **userFollowersNum** | **url** |
| 1 | torvalds | 52722 | https://github.com/torvalds |
| 2 | JakeWharton | 30161 | https://github.com/JakeWharton |
| 3 | Tj | 25827 | https://github.com/Tj |
| 4 | addyosmani | 24604 | https://github.com/addyosmani |
| 5 | paulirish | 24510 | https://github.com/paulirish |
| 6 | mojombo | 23076 | https://github.com/mojombo |
| 7 | defunkt | 18522 | https://github.com/defunkt |
| 8 | sindresorhus | 15755 | https://github.com/sindresorhus |
| 9 | douglascrockford | 15396 | https://github.com/douglascrockford |
| 10 | mbostock | 14928 | https://github.com/mbostock |
| 11 | jeresig | 14703 | https://github.com/jeresig |
| 12 | ruanyf | 14701 | https://github.com/ruanyf |
| 13 | daimajia | 14507 | https://github.com/daimajia |
| 14 | mattt | 13705 | https://github.com/mattt |
| 15 | mdo | 13443 | https://github.com/mdo |
| 16 | kennethreitz | 13360 | https://github.com/kennethreitz |
| 17 | schacon | 13102 | https://github.com/schacon |
| 18 | gaearon | 13007 | https://github.com/gaearon |
| 19 | jlord | 11897 | https://github.com/jlord |
| 20 | visionmedia | 11714 | https://github.com/visionmedia |

### 5.1.2 UserRank算法

基于用户粉丝数量的用户影响力分析算法虽然简单直接，能够在一定程度上体现用户影响力，但是该方法没有考虑每个关注链接的质量差异。如下图，用户a和用户b都拥有3个粉丝，其中a的粉丝的粉丝数都为0，而b的粉丝都有一些其他粉丝，那么显然，关注b的链接具有较高的价值。这种情况下，因为用户a和用户b有相同数量的粉丝数而认为他们两者具有相同的影响力显然不妥。针对这个问题，我们得出这样的假设：

高影响力用户关注的用户，其影响力也一定很高。

因此，我们基于PageRank算法提出了Github用户影响力分析的UserRank算法。



UserRank的公式如下：

其中，u为待估算影响力的用户，v为关注用户u的用户，Out(v)为用户v的出度，即用户v关注用户的数量，N为所有用户数量，q含义同上。



**算法 2：**

输入：Github社交网络有向关系图G=<V, E>；

输出：用户影响力度量以及排序

**算法步骤：**

1. 初始化用户UserRank列表L，用来存储UserRank度量评分值，初始化为1；
2. Map1存储用户以及关注用户的列表，即图中节点和指向该节点的节点列表；Map2存储用户以及其关注的用户数量，即图中节点以及其出度数；
3. 根据图G，遍历图中每一个节点，得到链入该节点的用户列表，并存储在Map1中；
4. 根据图G，遍历图中每一个节点，得到其出度数，并存储在Map2中；
5. 根据UserRank公式计算每个节点的UserRank评分，并存储在列表L中；
6. 将列表L按度量值降序排列；
7. 返回列表L；

### 5.1.3 HITS算法

## 5.2 Watcher

Github是一个基于项目托管平台Git的网络，用户和项目是该网络的核心。用户的影响力，除了体现在社交网络中吸引力大量粉丝，其拥有的项目受关注程度也是其影响力的一部分。用户的项目受到更多的关注，说明其项目影响力越大，那么用户本身通过项目产生的影响力也越大。

基本假设：一个项目受关注（watch）程度越高，说明该项目具有更大价值和影响力，其拥有者也就可以通过项目产生更大影响。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| rank | login | Hindex | AveNum | MaxNum | SumNum |
| 1 | sindresorhus | 128 | 168.86 | 50113 | 171900 |
| 2 | substack | 97 | 71.88 | 11262 | 67565 |
| 3 | Tj | 94 | 347.76 | 9688 | 101197 |
| 4 | visionmedia | 72 | 272.74 | 10112 | 60820 |
| 5 | maxogden | 62 | 44.44 | 4370 | 30929 |
| 6 | mafintosh | 62 | 41.82 | 3358 | 25010 |
| 7 | Firebase | 60 | 185.49 | 1651 | 24856 |
| 8 | tpope | 58 | 383.76 | 5803 | 31468 |
| 9 | Spatie | 58 | 104.31 | 1186 | 14082 |
| 10 | JakeWharton | 54 | 660.42 | 12825 | 64721 |
| 11 | addyosmani | 53 | 140.66 | 10969 | 41212 |
| 12 | nicklockwood | 52 | 400.50 | 7641 | 33642 |
| 13 | docker | 52 | 781.77 | 40002 | 86776 |
| 14 | feross | 51 | 118.98 | 10467 | 34146 |
| 15 | ccoenraets | 49 | 78.44 | 1367 | 10354 |
| 16 | Esri | 48 | 22.32 | 559 | 9441 |
| 17 | mattn | 47 | 13.83 | 1986 | 15465 |
| 18 | keijiro | 47 | 24.61 | 377 | 9006 |
| 19 | hadley | 46 | 61.78 | 1790 | 13776 |
| 20 | soffes | 45 | 70.11 | 2527 | 12900 |

## 5.3 Fork

作为开源代码库以及分布式版本控制系统，可以简易地“fork”项目，是Github的核心亮点和独特卖点之一。项目的派生（Fork）和拉拽请求（Pull request）构成Github最独具一格的工作模式。用户可以很方便地复制项目到自己账户独立使用并为项目贡献代码。首先用户找到自己希望参与的项目，并对派生项目的版本库具有读写的完全权限，就好像这个项目原本就是由自己创立的那样。当用户完成开发并向自己派生的版本库推送后，就可以向项目原来的拥有者发送一个Pull Request，请求审核。项目拥有者收到Pull Request后审核代码，审核通过后就可以直接执行合并操作接受贡献者的修改。

在Github进行分支就像在Facebook进行交友一样，在社会关系图的节点中不断的连线。所以我们对用户项目的fork情况进行分析。

统计用户创建的项目被fork的总次数、平均次数和被最多fork的项目的次数，以及用H-因子，即最多fork H次的至少有H个项目。结果如表所示：

同时分析，项目被fork后有多少项目向原来的项目提出了pull\_request，其中多少被merged。

在Github所有项目中，其中fork而来的项目有。。。，占总项目的。。。。在这些Fork而来的项目中，fork之后向原项目提出pull\_request的占。。。，其中有。。。被merged。

有。。。用户fork了别人的项目，约占总用户的。。。%。其中fork之后对原项目提出pull\_request的用户有。。。，占比例。。。%，其中有。。。%的用户有被merged的pull\_request。

## 5.4 Activity

Github网站中，通过统计用户创建commit数量、创建项目数量、open ISSUE 数量，open pull request数量来统计用户过去的贡献量，用户的这几种行为是用户在网站中的主要行为，我们用同样的标准来衡量用户的活跃度。

我们统计用户在近三个月（截止到数据收集日期）的用户活动量，并根据用户活动量为其排序。结果如下：

## 5.5 Correlation

以上四节我们分别从不同的角度对所有用户的不同影响力表现进行了统计和分析。本节，我们比较不同用户影响力度量方式得到的用户排名不同，而不是比较得到的度量值。我们按不同的度量对用户进行排序，排名为1的用户表明该角度度量其影响力最高。具有相同度量值的用户取其最高排名（引用使用相同方法的文章，否则取平均值引用Buck, 1980）。

我们使用Spearman’s 等级相关系数作为衡量不同度量排序之间相关性的标准。

这里xi和yi分别为用户在两个不同排序列表中的排名次序，N为排名列表中的总用户数。Spearman等级相关系数是无参的。The coefficient assesses how well an arbitrary monotonic function could describe the relationship between two variables, without making any other assumptions about the particular nature of the relationship between the variables. Our inclusive and complete dataset guarantees reliability of the correlation estimates. The range of *p* is from +1to -1. A perfect positive correlation is +1, a perfect negative correlation is -1, and no correlation is 0.



In order to eliminate the possibility that these correlations are due to the large numbers of github members with one or more “zero” values, we performed a more focused analysis, examining only the relatively influential users. 我们分别基于不同度量标准取排序列表的top 10%, top 1%, top 1000, top 100. 然后成对的计算不同排序之间的相关系数，结果如下表。

1. http://www.ghtorrent.org/downloads.html [↑](#footnote-ref-0)