# Python网络编程期末作业

##### 题目：基于python网络爬虫获取江西师范大学官方微博的数据并进 行分析

###### 学生姓名：XX

###### 学号：XX

###### 班级：16级XXX

目录

[一． 目的和意义 2](#_Toc26805_WPSOffice_Level1)

[二． 使用的技术和框架 2](#_Toc18157_WPSOffice_Level1)

[2.1 网络请求部分 2](#_Toc18157_WPSOffice_Level2)

[2.2 数据解析部分 3](#_Toc28644_WPSOffice_Level2)

[2.3 数据存储部分 3](#_Toc5161_WPSOffice_Level2)

[2.4 数据可视化部分 3](#_Toc10442_WPSOffice_Level2)

[三． 实现过程 4](#_Toc28644_WPSOffice_Level1)

[3.1 前期准备 4](#_Toc2505_WPSOffice_Level2)

[3.2 数据获取 4](#_Toc24723_WPSOffice_Level2)

[3.3 数据的存储 5](#_Toc11151_WPSOffice_Level2)

[3.4 数据的可视化 6](#_Toc26240_WPSOffice_Level2)

[四． 代码实现 7](#_Toc5161_WPSOffice_Level1)

[4.1 weibo\_spider.py文件 7](#_Toc1075_WPSOffice_Level2)

[4.2 weibo\_db.py文件 16](#_Toc6386_WPSOffice_Level2)

[4.3 anlyze.py文件 22](#_Toc17626_WPSOffice_Level2)

[五． 结果分析 31](#_Toc10442_WPSOffice_Level1)

[5.1 用户的社交关系分析 31](#_Toc10149_WPSOffice_Level2)

[5.2 对粉丝群体的分析 35](#_Toc7404_WPSOffice_Level2)

[5.3 对发布的微博进行分析 38](#_Toc12431_WPSOffice_Level2)

[六. 总结 40](#_Toc2505_WPSOffice_Level1)

### 目的和意义

如今,人们已经习惯于通过社交网络来表达观点和情感,并与其他人进行互动,这其中蕴藏着许多有价值的信息。而社交网络的本质是将现实世界的人与人之间的联系映射到虚拟空间,用户关系显然是最重要的。由于社交网络数据本身的商业价值,研究人员很难直接从互联网公司获取。因此,本文以目前最流行的社交网络平台——新浪微博为基础,获取江西师范大学大学官方微博的数据,设计并实现了用户关系可视化及数据的统计分析,在一定程度上解决了上述问题。

### 使用的技术和框架

#### 2.1 网络请求部分

网络请求部分使用的是Requests 库。Requests 是用[Python](http://blog.csdn.net/shanzhizi)语言编写，基于 urllib，采用 Apache2 Licensed 开源协议的 HTTP 库。它比 urllib 更加方便，可以节约我们大量的工作。

#### 2.2 数据解析部分

数据解析部分使用的是python标准库里的json模块,因为发送的数据是ajax请求，所以返回的json数据直接用json模块解析即可，并没有使用beautifulsoup等第三方解析库。

#### 2.3 数据存储部分

数据存储部分使用mysql数据库,在程序中导入pymysql模块通过代码对mysql数据库进行操作。Mysql是一个小型关系型数据库管理系统,mysql被广泛地应用在Internet上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，许多中小型网站为了降低网站总体拥有成本而选择了mysql作为网站数据库。

#### 2.4 数据可视化部分

人物关系的数据可视化部分使用的是neo4j图形数据库，并使用了py2neo模块对数据库进行操作。[Neo4j](https://baike.baidu.com/item/Neo4j)是一个高性能的NOSQL图形数据库，它将结构化数据存储在网络上而不是表中。它是一个[嵌入式](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F/575465)的、基于[磁盘](https://baike.baidu.com/item/%E7%A3%81%E7%9B%98/2842227)的、具备完全的事务特性的Java持久化引擎，它将结构化数据存储在网络(从数学角度叫做图)上而不是表中。当图的结构建立完成后可以通过cypher语句轻松获取感兴趣的数据部分并展示。

统计信息的可视化部分使用的是matplotlib库。Matplotlib是一个Python的2D绘图库，通过Matplotlib，开发者可以仅需要几行代码，便可以生成绘图。一般可绘制折线图、散点图、柱状图、饼图、直方图、子图等等。

### 实现过程

#### 3.1 前期准备

在编写代码实现之前首先对weibo的站点结构进行了分析，发现微博有weibo.cn m.weibo.com [www.weibo.com三个站点，我选择的是通过m.weibo.com移动版的站点来获取数据。然后就是通过selenium来实现自动登录并保存cookies,后续的request请求都带上这个cookies，使用浏览器的开发者工具可以发现通过发送ajax请求可以获取用户的个人信息，关注列表，粉丝列表以及微博列表，在完成了自动登录代码后还发现这几个url根本就不需要登录即可返回正确的数据，由于不需要登录，所以不存在封号的风险，但请求太频繁会返回418状态码并禁止访问，等待一分钟左右便可恢复访问。](http://www.weibo.com三个站点，我选择的是通过移动版的站点来获取数据。然后就是通过selenium来实现自动登录并保存cookies,后续的request请求都带上这个cookies，使用浏览器的开发者工具可以发现通过发送ajax请求可以获取用户的个人信息，关注列表，粉丝列表以及微博列表，在完成了自动登录代码后还发现这几个url根本就不需要登录即可返回正确的数据，由于不需要登录，所以不存在封号的风险，但请求太频繁会返回418状态码并禁止访问，等待一分钟左右便可恢复访问。)

#### 3.2 数据获取

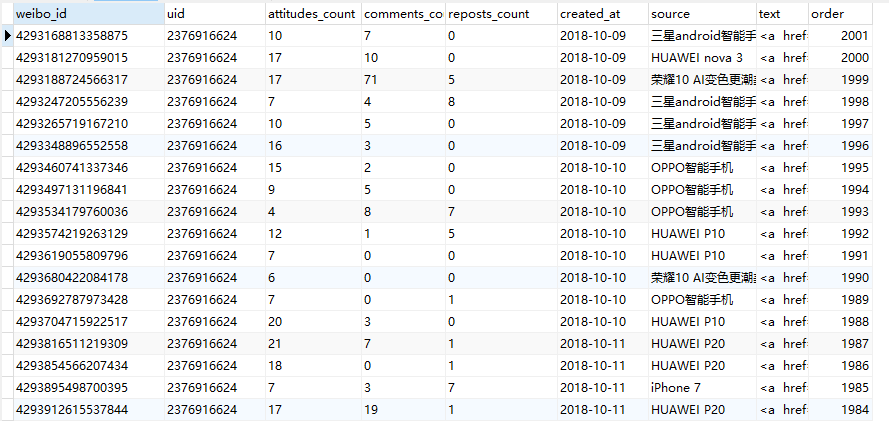
程序开始运行时先从指定的用户(及江西师范大学官方微博)开始访问，先获取到该用户的·个人信息，然后再访问此用户的关注列表中的用户及粉丝列表中的用户的uid，将这些用户的uid与该用户的关系存储到关系表中，待所以关系获取完成后再从数据库中取出这些uid并逐个获取他们的详细用户信息，然后再将这些信息存储到用户表中，这样就获得了用户的信息及关系。这些爬完之后就接着继续爬取用户的微博列表并存入到微博表中。

#### 3.3 数据的存储

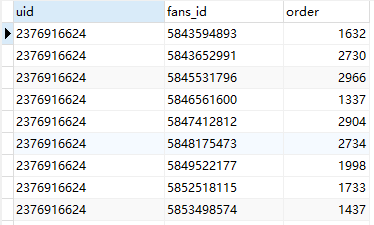
获取到的数据通过json模块解析出感兴趣的部分后持久化到mysql相应的标注，表的结构如下：

其中fans\_relation\_table和follow\_relation\_table用来存储用户的关系，通过uid将他们关联到一起。user\_table和weibo\_table分别用来存储用户信息和微博信息，通过关联表可以方便的将他们连接到一起。comments\_table用于存储评论信息，由于时间原因暂时没有用到，留到这里方便以后扩展。在代码中加了一些防重爬措施，可以为补爬数据提供方便，增加了程序的扩展性。代码中也添加了很多异常处理，提升了程序的健壮性。

一些数据库的截图：





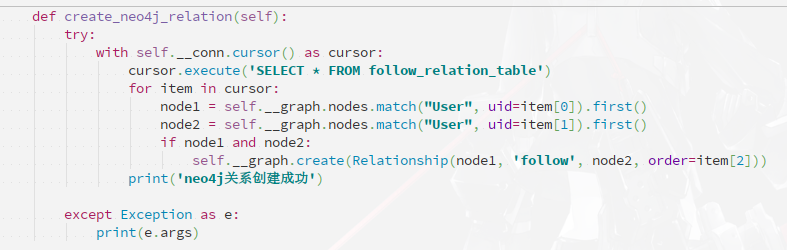


#### 3.4 数据的可视化

人物关系可视化是通过将mysql数据库中的数据转存到neo4j图形数据库中，再neo4j中构造出图的结构，然后再通过cypher语句来查找并显示的。

数据转存相关代码：





先从用户信息表中获取信息构造出所有的点，再从关系表中构造出所有的关系即生成了用户关系图的结构。

数据统计分析部分使用sql语句查询聚合等操作，并加上python代码的逻辑处理提供数据给matplotlib渲染并显示。

### 代码实现

#### 4.1 weibo\_spider.py文件

import matplotlib  
import pymysql  
import numpy as np  
import matplotlib.pyplot as plt  
  
from py2neo import Graph, Node, Relationship  
  
matplotlib.rcParams[**'font.sans-serif'**] = [**'SimHei'**]  
matplotlib.rcParams[**'axes.unicode\_minus'**] = False  
  
  
class Analyze:  
 def \_\_init\_\_(self):  
 # 连接mysql数据库  
 self.\_\_conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**'root'**, passwd=**'123'**, db=**'weibo'**,  
 charset=**'utf8'**)  
 print(**'mysql连接成功'**)  
 # 连接neo4j数据库  
 self.\_\_graph = Graph(**'http://localhost:7474'**, username=**'neo4j'**, password=**'123'**)  
 print(**'neo4j连接成功'**)  
 # self.\_\_graph.delete\_all()  
  
 def create\_neo4j\_node(self, uid):  
 sql1 = **'SELECT \* FROM user\_table WHERE user\_table.uid = {}'**.format(uid)  
 sql2 = **'''  
 SELECT  
 user\_table.uid,  
 user\_table.`name`,  
 user\_table.gender,  
 user\_table.verified,  
 user\_table.verified\_reason,  
 user\_table.urank,  
 user\_table.mbrank,  
 user\_table.weibos\_count,  
 user\_table.fans\_count,  
 user\_table.follows\_count,  
 user\_table.description  
 FROM  
 user\_table ,  
 follow\_relation\_table  
 WHERE  
 follow\_relation\_table.uid = {} AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 '''**.format(uid)  
 try:  
 with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql1)  
 user = cursor.fetchone()  
 self.\_\_graph.create(Node(**'User'**, uid=user[0], name=user[1], gender=user[2], verified=user[3],  
 verified\_reason=user[4], urank=user[5], mbrank=user[6],  
 weibos\_count=user[7], fans\_count=user[8], follows\_count=user[9],  
 description=user[10]))  
 cursor.execute(sql2)  
 for item in cursor:  
 self.\_\_graph.create(Node(**'User'**, uid=item[0], name=item[1], gender=item[2], verified=item[3],  
 verified\_reason=item[4], urank=item[5], mbrank=item[6],  
 weibos\_count=item[7], fans\_count=item[8], follows\_count=item[9],  
 description=item[10]))  
 print(**'neo4j节点创建成功'**)  
  
 except Exception as e:  
 print(e.args)  
  
 def create\_neo4j\_relation(self):  
 try:  
 with self.\_\_conn.cursor() as cursor:  
 cursor.execute(**'SELECT \* FROM follow\_relation\_table'**)  
 for item in cursor:  
 node1 = self.\_\_graph.nodes.match(**"User"**, uid=item[0]).first()  
 node2 = self.\_\_graph.nodes.match(**"User"**, uid=item[1]).first()  
 if node1 and node2:  
 self.\_\_graph.create(Relationship(node1, **'follow'**, node2, order=item[2]))  
 print(**'neo4j关系创建成功'**)  
  
 except Exception as e:  
 print(e.args)  
  
 def analyz\_follow\_sex\_ratio(self):  
 sql = **'''SELECT  
 user\_table.gender,count(user\_table.gender)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.gender'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'男'**, **'女'**], [dic[**'m'**], dic[**'f'**]], [0.05, 0], **'江西师范大学官方微博关注用户的男女比例'**)  
  
 def analyz\_fans\_sex\_ratio(self):  
 sql = **'''SELECT  
 user\_table.gender,count(user\_table.gender)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.gender'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'男'**, **'女'**], [dic[**'m'**], dic[**'f'**]], [0.05, 0], **'江西师范大学官方微博粉丝的男女比例'**)  
  
 def analyz\_fans\_urank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.urank,count(user\_table.urank)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.urank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0]  
 for item in cursor:  
 if 0 <= item[0] < 10:  
 li[0] += item[1]  
 elif 10 <= item[0] < 20:  
 li[1] += item[1]  
 elif 20 <= item[0] < 30:  
 li[2] += item[1]  
 elif 30 <= item[0] < 40:  
 li[3] += item[1]  
 elif 40 <= item[0] <= 48:  
 li[4] += item[1]  
 self.draw\_bar([**'0-9级'**, **'10-19级'**, **'20-19级'**, **'30-39级'**, **'40-48级'**], li, **'人数'**, **'江西师范大学官方微博粉丝的用户等级分布'**)  
  
 def analyz\_fans\_mbrank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.mbrank,count(user\_table.mbrank)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.mbrank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0, 0, 0]  
 for item in cursor:  
 li[item[0]] += item[1]  
 self.draw\_bar([**'0级'**, **'1级'**, **'2级'**, **'3级'**, **'4级'**, **'5级'**, **'6级'**], li, **'人数'**, **'江西师范大学官方微博粉丝的会员等级分布'**)  
  
 def analyz\_follows\_urank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.urank,count(user\_table.urank)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.urank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0]  
 for item in cursor:  
 if 0 <= item[0] < 10:  
 li[0] += item[1]  
 elif 10 <= item[0] < 20:  
 li[1] += item[1]  
 elif 20 <= item[0] < 30:  
 li[2] += item[1]  
 elif 30 <= item[0] < 40:  
 li[3] += item[1]  
 elif 40 <= item[0] <= 48:  
 li[4] += item[1]  
 self.draw\_bar([**'0-9级'**, **'10-19级'**, **'20-19级'**, **'30-39级'**, **'40-48级'**], li, **'人数'**, **'江西师范大学官方微博关注的用户等级分布'**)  
  
 def analyz\_follows\_mbrank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.mbrank,count(user\_table.mbrank)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.mbrank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0, 0, 0, 0]  
 for item in cursor:  
 li[item[0]] += item[1]  
 print(li)  
 self.draw\_bar([**'0级'**, **'1级'**, **'2级'**, **'3级'**, **'4级'**, **'5级'**, **'6级'**, **'7级'**], li, **'人数'**, **'江西师范大学官方微博关注用户的会员等级分布'**)  
  
 def analyz\_fans\_verified\_ratio(self):  
 sql = **'''SELECT  
 user\_table.verified,count(user\_table.verified)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.verified'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'未认证'**, **'已认证'**], [dic[**'0'**], dic[**'1'**]], [0.05, 0], **'江西师范大学官方微博粉丝的认证比例'**)  
  
 def analyz\_follows\_verified\_ratio(self):  
 sql = **'''SELECT  
 user\_table.verified,count(user\_table.verified)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.verified'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'未认证'**, **'已认证'**], [dic[**'0'**], dic[**'1'**]], [0.05, 0], **'江西师范大学官方微博关注用户的认证比例'**)  
  
 def analyz\_weibo\_post\_tendency(self):  
 sql = **'''  
 SELECT  
 weibo\_table.created\_at,  
 count(weibo\_table.created\_at)  
 FROM  
 weibo\_table  
 WHERE  
 weibo\_table.uid = 2376916624  
 GROUP BY  
 weibo\_table.created\_at  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 x = [**'2018-10'**, **'2018-11'**, **'2018-12'**, **'2019-01'**, **'2019-02'**, **'2019-03'**, **'2019-04'**, **'2019-05'**, **'2019-06'**]  
 y = [0, 0, 0, 0, 0, 0, 0, 0, 0]  
 dic = dict(zip(x, y))  
 for item in cursor:  
 dic[item[0][:7]] += item[1]  
 xticks, heights = [], []  
 print(dic)  
 for k, v in dic.items():  
 xticks.append(k)  
 heights.append(v)  
  
 self.draw\_plot(xticks, heights, **'发布总数'**, **'江西师范大学官方微博每月发布微博数变化趋势图'**)  
  
 def analyz\_weibo\_source(self):  
 sql = **'''SELECT  
 weibo\_table.source,  
 count(weibo\_table.source)  
 FROM  
 weibo\_table  
 WHERE  
 weibo\_table.uid = 2376916624  
 GROUP BY  
 weibo\_table.source  
 HAVING  
 source!=''  
 ORDER BY COUNT(weibo\_table.source) DESC  
 LIMIT 10'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 results = cursor.fetchall()  
 source\_list = []  
 count\_list = []  
 for source, count in results:  
 source\_list.append(source)  
 count\_list.append(count)  
  
 name\_list = source\_list  
 num\_list = count\_list  
 plt.barh(range(len(num\_list)), num\_list, color=**'rgb'**, tick\_label=name\_list)  
 plt.title(**'使用次数排名前10的发布微博设备'**)  
 plt.show()  
  
 def draw\_pie(self, labels, sizes, explode, title):  
 plt.pie(sizes, labels=labels, autopct=**'%1.1f%%'**, shadow=True, explode=explode, startangle=50)  
 plt.axis(**'equal'**)  
 plt.title(title)  
 plt.show()  
  
 def draw\_bar(self, xticks, heights, ylabel, title):  
 plt.bar(range(len(heights)), heights, align=**'center'**, color=**'steelblue'**, alpha=0.8)  
 plt.ylabel(ylabel)  
 plt.title(title)  
 plt.xticks(range(len(xticks)), xticks)  
 for x, y in enumerate(heights):  
 plt.text(x, y, **'%s'** % round(y, 1), ha=**'center'**)  
 plt.show()  
  
 def draw\_plot(self, xticks, heights, ylabel, title):  
 plt.plot(xticks, heights)  
 plt.ylabel(ylabel)  
 plt.title(title)  
 plt.show()  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 analyze = Analyze()  
 analyze.create\_neo4j\_node(**'2376916624'**)  
 analyze.create\_neo4j\_relation()  
 analyze.draw\_pie([**'A'**, **'B'**], [2, 3], [0.05, 0], **'test'**)  
 analyze.analyz\_follow\_sex\_ratio()  
 analyze.analyz\_fans\_sex\_ratio()  
 analyze.analyz\_fans\_urank\_distribution()  
 analyze.analyz\_fans\_mbrank\_distribution()  
 analyze.analyz\_follows\_urank\_distribution()  
 analyze.analyz\_follows\_mbrank\_distribution()  
 analyze.analyz\_fans\_verified\_ratio()  
 analyze.analyz\_follows\_verified\_ratio()  
 analyze.analyz\_weibo\_source()  
 analyze.analyz\_weibo\_post\_tendency()

#### 4.2 weibo\_db.py文件

import pymysql  
  
  
class DbHelper:  
  
 def \_\_init\_\_(self):  
 self.create\_user\_table\_sql = **'''  
 CREATE TABLE `user\_table` (  
 `uid` varchar(255) NOT NULL,  
 `name` varchar(255) NOT NULL,  
 `gender` varchar(255) NOT NULL,  
 `verified` varchar(255) NOT NULL,  
 `verified\_reason` varchar(255) DEFAULT NULL,  
 `urank` int(11) NOT NULL,  
 `mbrank` int(11) NOT NULL,  
 `weibos\_count` int(11) NOT NULL,  
 `fans\_count` int(11) NOT NULL,  
 `follows\_count` int(11) NOT NULL,  
 `description` varchar(255) DEFAULT NULL,  
 PRIMARY KEY (`uid`)  
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  
 '''** self.create\_follow\_relation\_table\_sql = **'''  
 CREATE TABLE `follow\_relation\_table` (  
 `uid` varchar(255) NOT NULL,  
 `follow\_id` varchar(255) NOT NULL,  
 PRIMARY KEY (`uid`,`follow\_id`) USING BTREE  
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  
 '''** self.create\_fans\_relation\_table\_sql = **'''  
 CREATE TABLE `fans\_relation\_table` (  
 `uid` varchar(255) NOT NULL,  
 `fans\_id` varchar(255) NOT NULL,  
 PRIMARY KEY (`uid`,`fans\_id`) USING BTREE  
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  
 '''** self.create\_weibo\_table\_sql = **'''  
 CREATE TABLE `weibo\_table` (  
 `weibo\_id` varchar(255) NOT NULL,  
 `uid` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 `attitudes\_count` varchar(255) NOT NULL,  
 `comments\_count` varchar(255) NOT NULL,  
 `reposts\_count` varchar(255) NOT NULL,  
 `created\_at` varchar(255) NOT NULL,  
 `source` varchar(255) DEFAULT NULL,  
 `text` varchar(8196) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 PRIMARY KEY (`weibo\_id`)  
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  
 '''** self.create\_comments\_table\_sql = **'''  
 CREATE TABLE `comments\_table` (  
 `comment\_id` varchar(255) NOT NULL,  
 `uid` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 `weibo\_id` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 `created\_at` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 `source` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci DEFAULT NULL,  
 `text` varchar(1024) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 `like\_counts` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL,  
 PRIMARY KEY (`comment\_id`)  
 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;  
 '''** self.insert\_user\_table\_sql = **'''  
 INSERT INTO `weibo`.`user\_table`(`uid`, `name`, `gender`, `verified`,   
 `verified\_reason`, `urank`, `mbrank`, `weibos\_count`, `fans\_count`, `follows\_count`, `description`) VALUES (  
 %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);   
 '''** self.insert\_follow\_relation\_table\_sql = **'''  
 INSERT INTO `weibo`.`follow\_relation\_table`(`uid`, `follow\_id`, `order`) VALUES (%s, %s, %s);  
 '''** self.insert\_fans\_relation\_table\_sql = **'''  
 INSERT INTO `weibo`.`fans\_relation\_table`(`uid`, `fans\_id`, `order`) VALUES (%s, %s, %s);  
 '''** self.insert\_weibo\_table\_sql = **'''  
 INSERT INTO `weibo`.`weibo\_table`(`weibo\_id`, `uid`, `attitudes\_count`,   
 `comments\_count`, `reposts\_count`, `created\_at`, `source`, `text`, `order`) VALUES (%s, %s, %s, %s, %s,   
 %s, %s, %s, %s);   
 '''** self.insert\_comments\_table\_sql = **'''  
 INSERT INTO `weibo`.`comments\_table`(`comment\_id`, `uid`, `weibo\_id`,   
 `created\_at`, `source`, `text`, `like\_counts`, `order`) VALUES (%s, %s, %s, %s, %s, %s, %s, %s);   
 '''** self.conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**'root'**, passwd=**'123'**, db=**'weibo'**, charset=**'utf8'**)  
 print(**'mysql连接成功'**)  
  
 def \_\_del\_\_(self):  
 self.conn.close()  
  
 def create\_all\_table(self):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(DbHelper.create\_user\_table\_sql)  
 cursor.execute(DbHelper.create\_follow\_relation\_table\_sql)  
 cursor.execute(DbHelper.create\_fans\_relation\_table\_sql)  
 cursor.execute(DbHelper.create\_weibo\_table\_sql)  
 cursor.execute(DbHelper.create\_comments\_table\_sql)  
 print(**'所有创建表成功'**)  
 except Exception as e:  
 print(e.args)  
  
 def save\_to\_user\_table(self, user):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(self.insert\_user\_table\_sql, (  
 user.uid, user.name, user.gender, user.verified, user.verified\_reason,  
 user.urank, user.mbrank, user.weibos\_count,  
 user.fans\_count, user.follows\_count, user.description  
 ))  
 self.conn.commit()  
 except Exception as e:  
 print(e.args)  
  
 def save\_to\_weibo\_table(self, weibo):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(self.insert\_weibo\_table\_sql, (  
 weibo.weibo\_id, weibo.uid, weibo.attitudes\_count, weibo.comments\_count, weibo.reposts\_count,  
 weibo.created\_at, weibo.source, weibo.text, weibo.order  
 ))  
 self.conn.commit()  
 except Exception as e:  
 print(e.args)  
  
 def save\_to\_follow\_relation\_table(self, follow\_relation):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(self.insert\_follow\_relation\_table\_sql,  
 (follow\_relation.uid, follow\_relation.follow, follow\_relation.order))  
 self.conn.commit()  
 except Exception as e:  
 print(e.args)  
  
 def save\_to\_fans\_relation\_table(self, fans\_relation):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(self.insert\_fans\_relation\_table\_sql,  
 (fans\_relation.uid, fans\_relation.fans, fans\_relation.order))  
 self.conn.commit()  
 except Exception as e:  
 print(e.args)  
  
 def save\_to\_comments\_table(self, comment):  
 try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(self.insert\_comments\_table\_sql, (  
 comment.comment\_id, comment.uid, comment.weibo\_id, comment.created\_at, comment.source, comment.text,  
 comment.like\_counts, comment.order))  
 self.conn.commit()  
 except Exception as e:  
 print(e.args)  
  
 def get\_fans\_uid(self, uid):  
 select\_sql = **'''SELECT DISTINCT  
 fans\_relation\_table.fans\_id  
 FROM  
 fans\_relation\_table  
 WHERE  
 fans\_relation\_table.uid = %s  
 '''** try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(select\_sql, uid)  
 yield from cursor.fetchall()  
 except Exception as e:  
 print(e.args)  
  
 def get\_follow\_uid(self, uid):  
 select\_sql = **'''SELECT DISTINCT  
 follow\_relation\_table.follow\_id  
 FROM  
 follow\_relation\_table  
 WHERE  
 follow\_relation\_table.uid = %s  
 '''** try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(select\_sql, uid)  
 yield from cursor.fetchall()  
 except Exception as e:  
 print(e.args)  
  
 def get\_crawled\_uid(self):  
 select\_sql = **'''SELECT  
 user\_table.uid  
 FROM  
 user\_table  
 '''** try:  
 with self.conn.cursor() as cursor:  
 cursor.execute(select\_sql)  
 yield from cursor.fetchall()  
 except Exception as e:  
 print(e.args)  
  
  
class User:  
 def \_\_init\_\_(self, uid, name, gender, verified, verified\_reason, urank, mbrank, weibos\_count, fans\_count,  
 follows\_count, description):  
 self.uid = uid  
 self.name = name  
 self.gender = gender  
 self.verified = verified  
 self.verified\_reason = verified\_reason  
 self.urank = urank  
 self.mbrank = mbrank  
 self.weibos\_count = weibos\_count  
 self.fans\_count = fans\_count  
 self.follows\_count = follows\_count  
 self.description = description  
  
  
class FollowRelation:  
 def \_\_init\_\_(self, uid, follow, order):  
 self.uid = uid  
 self.follow = follow  
 self.order = order  
  
  
class FansRelation:  
 def \_\_init\_\_(self, uid, fans, order):  
 self.uid = uid  
 self.fans = fans  
 self.order = order  
  
  
class Weibo:  
 def \_\_init\_\_(self, uid, weibo\_id, attitudes\_count, comments\_count, reposts\_count, created\_at, source, text, order):  
 self.uid = uid  
 self.weibo\_id = weibo\_id  
 self.attitudes\_count = attitudes\_count  
 self.comments\_count = comments\_count  
 self.reposts\_count = reposts\_count  
 self.created\_at = created\_at  
 self.source = source  
 self.text = text  
 self.order = order  
  
  
class Comment:  
 def \_\_init\_\_(self, uid, weibo\_id, comment\_id, created\_at, source, text, like\_counts, order):  
 self.uid = uid  
 self.weibo\_id = weibo\_id  
 self.comment\_id = comment\_id  
 self.created\_at = created\_at  
 self.source = source  
 self.text = text  
 self.like\_counts = like\_counts  
 self.order = order

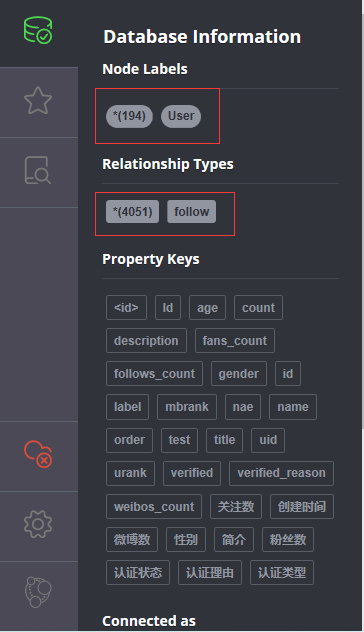
#### 4.3 anlyze.py文件

import matplotlib  
import pymysql  
import numpy as np  
import matplotlib.pyplot as plt  
  
from py2neo import Graph, Node, Relationship  
  
matplotlib.rcParams[**'font.sans-serif'**] = [**'SimHei'**]  
matplotlib.rcParams[**'axes.unicode\_minus'**] = False  
  
  
class Analyze:  
 def \_\_init\_\_(self):  
 # 连接mysql数据库  
 self.\_\_conn = pymysql.connect(host=**'127.0.0.1'**, port=3306, user=**'root'**, passwd=**'123'**, db=**'weibo'**,  
 charset=**'utf8'**)  
 print(**'mysql连接成功'**)  
 # 连接neo4j数据库  
 self.\_\_graph = Graph(**'http://localhost:7474'**, username=**'neo4j'**, password=**'123'**)  
 print(**'neo4j连接成功'**)  
 # self.\_\_graph.delete\_all()  
  
 def create\_neo4j\_node(self, uid):  
 sql1 = **'SELECT \* FROM user\_table WHERE user\_table.uid = {}'**.format(uid)  
 sql2 = **'''  
 SELECT  
 user\_table.uid,  
 user\_table.`name`,  
 user\_table.gender,  
 user\_table.verified,  
 user\_table.verified\_reason,  
 user\_table.urank,  
 user\_table.mbrank,  
 user\_table.weibos\_count,  
 user\_table.fans\_count,  
 user\_table.follows\_count,  
 user\_table.description  
 FROM  
 user\_table ,  
 follow\_relation\_table  
 WHERE  
 follow\_relation\_table.uid = {} AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 '''**.format(uid)  
 try:  
 with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql1)  
 user = cursor.fetchone()  
 self.\_\_graph.create(Node(**'User'**, uid=user[0], name=user[1], gender=user[2], verified=user[3],  
 verified\_reason=user[4], urank=user[5], mbrank=user[6],  
 weibos\_count=user[7], fans\_count=user[8], follows\_count=user[9],  
 description=user[10]))  
 cursor.execute(sql2)  
 for item in cursor:  
 self.\_\_graph.create(Node(**'User'**, uid=item[0], name=item[1], gender=item[2], verified=item[3],  
 verified\_reason=item[4], urank=item[5], mbrank=item[6],  
 weibos\_count=item[7], fans\_count=item[8], follows\_count=item[9],  
 description=item[10]))  
 print(**'neo4j节点创建成功'**)  
  
 except Exception as e:  
 print(e.args)  
  
 def create\_neo4j\_relation(self):  
 try:  
 with self.\_\_conn.cursor() as cursor:  
 cursor.execute(**'SELECT \* FROM follow\_relation\_table'**)  
 for item in cursor:  
 node1 = self.\_\_graph.nodes.match(**"User"**, uid=item[0]).first()  
 node2 = self.\_\_graph.nodes.match(**"User"**, uid=item[1]).first()  
 if node1 and node2:  
 self.\_\_graph.create(Relationship(node1, **'follow'**, node2, order=item[2]))  
 print(**'neo4j关系创建成功'**)  
  
 except Exception as e:  
 print(e.args)  
  
 def analyz\_follow\_sex\_ratio(self):  
 sql = **'''SELECT  
 user\_table.gender,count(user\_table.gender)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.gender'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'男'**, **'女'**], [dic[**'m'**], dic[**'f'**]], [0.05, 0], **'江西师范大学官方微博关注用户的男女比例'**)  
  
 def analyz\_fans\_sex\_ratio(self):  
 sql = **'''SELECT  
 user\_table.gender,count(user\_table.gender)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.gender'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'男'**, **'女'**], [dic[**'m'**], dic[**'f'**]], [0.05, 0], **'江西师范大学官方微博粉丝的男女比例'**)  
  
 def analyz\_fans\_urank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.urank,count(user\_table.urank)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.urank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0]  
 for item in cursor:  
 if 0 <= item[0] < 10:  
 li[0] += item[1]  
 elif 10 <= item[0] < 20:  
 li[1] += item[1]  
 elif 20 <= item[0] < 30:  
 li[2] += item[1]  
 elif 30 <= item[0] < 40:  
 li[3] += item[1]  
 elif 40 <= item[0] <= 48:  
 li[4] += item[1]  
 self.draw\_bar([**'0-9级'**, **'10-19级'**, **'20-19级'**, **'30-39级'**, **'40-48级'**], li, **'人数'**, **'江西师范大学官方微博粉丝的用户等级分布'**)  
  
 def analyz\_fans\_mbrank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.mbrank,count(user\_table.mbrank)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.mbrank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0, 0, 0]  
 for item in cursor:  
 li[item[0]] += item[1]  
 self.draw\_bar([**'0级'**, **'1级'**, **'2级'**, **'3级'**, **'4级'**, **'5级'**, **'6级'**], li, **'人数'**, **'江西师范大学官方微博粉丝的会员等级分布'**)  
  
 def analyz\_follows\_urank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.urank,count(user\_table.urank)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.urank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0]  
 for item in cursor:  
 if 0 <= item[0] < 10:  
 li[0] += item[1]  
 elif 10 <= item[0] < 20:  
 li[1] += item[1]  
 elif 20 <= item[0] < 30:  
 li[2] += item[1]  
 elif 30 <= item[0] < 40:  
 li[3] += item[1]  
 elif 40 <= item[0] <= 48:  
 li[4] += item[1]  
 self.draw\_bar([**'0-9级'**, **'10-19级'**, **'20-19级'**, **'30-39级'**, **'40-48级'**], li, **'人数'**, **'江西师范大学官方微博关注的用户等级分布'**)  
  
 def analyz\_follows\_mbrank\_distribution(self):  
 sql = **'''  
 SELECT  
 user\_table.mbrank,count(user\_table.mbrank)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.mbrank  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 li = [0, 0, 0, 0, 0, 0, 0, 0]  
 for item in cursor:  
 li[item[0]] += item[1]  
 print(li)  
 self.draw\_bar([**'0级'**, **'1级'**, **'2级'**, **'3级'**, **'4级'**, **'5级'**, **'6级'**, **'7级'**], li, **'人数'**, **'江西师范大学官方微博关注用户的会员等级分布'**)  
  
 def analyz\_fans\_verified\_ratio(self):  
 sql = **'''SELECT  
 user\_table.verified,count(user\_table.verified)  
 FROM  
 fans\_relation\_table ,  
 user\_table  
 WHERE  
 fans\_relation\_table.uid = 2376916624 AND  
 fans\_relation\_table.fans\_id = user\_table.uid  
 GROUP BY  
 user\_table.verified'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'未认证'**, **'已认证'**], [dic[**'0'**], dic[**'1'**]], [0.05, 0], **'江西师范大学官方微博粉丝的认证比例'**)  
  
 def analyz\_follows\_verified\_ratio(self):  
 sql = **'''SELECT  
 user\_table.verified,count(user\_table.verified)  
 FROM  
 follow\_relation\_table ,  
 user\_table  
 WHERE  
 follow\_relation\_table.uid = 2376916624 AND  
 follow\_relation\_table.follow\_id = user\_table.uid  
 GROUP BY  
 user\_table.verified'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 dic = {}  
 for item in cursor:  
 dic[item[0]] = item[1]  
 self.draw\_pie([**'未认证'**, **'已认证'**], [dic[**'0'**], dic[**'1'**]], [0.05, 0], **'江西师范大学官方微博关注用户的认证比例'**)  
  
 def analyz\_weibo\_post\_tendency(self):  
 sql = **'''  
 SELECT  
 weibo\_table.created\_at,  
 count(weibo\_table.created\_at)  
 FROM  
 weibo\_table  
 WHERE  
 weibo\_table.uid = 2376916624  
 GROUP BY  
 weibo\_table.created\_at  
 '''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 x = [**'2018-10'**, **'2018-11'**, **'2018-12'**, **'2019-01'**, **'2019-02'**, **'2019-03'**, **'2019-04'**, **'2019-05'**, **'2019-06'**]  
 y = [0, 0, 0, 0, 0, 0, 0, 0, 0]  
 dic = dict(zip(x, y))  
 for item in cursor:  
 dic[item[0][:7]] += item[1]  
 xticks, heights = [], []  
 print(dic)  
 for k, v in dic.items():  
 xticks.append(k)  
 heights.append(v)  
  
 self.draw\_plot(xticks, heights, **'发布总数'**, **'江西师范大学官方微博每月发布微博数变化趋势图'**)  
  
 def analyz\_weibo\_source(self):  
 sql = **'''SELECT  
 weibo\_table.source,  
 count(weibo\_table.source)  
 FROM  
 weibo\_table  
 WHERE  
 weibo\_table.uid = 2376916624  
 GROUP BY  
 weibo\_table.source  
 HAVING  
 source!=''  
 ORDER BY COUNT(weibo\_table.source) DESC  
 LIMIT 10'''** with self.\_\_conn.cursor() as cursor:  
 cursor.execute(sql)  
 results = cursor.fetchall()  
 source\_list = []  
 count\_list = []  
 for source, count in results:  
 source\_list.append(source)  
 count\_list.append(count)  
  
 name\_list = source\_list  
 num\_list = count\_list  
 plt.barh(range(len(num\_list)), num\_list, color=**'rgb'**, tick\_label=name\_list)  
 plt.title(**'使用次数排名前10的发布微博设备'**)  
 plt.show()  
  
 def draw\_pie(self, labels, sizes, explode, title):  
 plt.pie(sizes, labels=labels, autopct=**'%1.1f%%'**, shadow=True, explode=explode, startangle=50)  
 plt.axis(**'equal'**)  
 plt.title(title)  
 plt.show()  
  
 def draw\_bar(self, xticks, heights, ylabel, title):  
 plt.bar(range(len(heights)), heights, align=**'center'**, color=**'steelblue'**, alpha=0.8)  
 plt.ylabel(ylabel)  
 plt.title(title)  
 plt.xticks(range(len(xticks)), xticks)  
 for x, y in enumerate(heights):  
 plt.text(x, y, **'%s'** % round(y, 1), ha=**'center'**)  
 plt.show()  
  
 def draw\_plot(self, xticks, heights, ylabel, title):  
 plt.plot(xticks, heights)  
 plt.ylabel(ylabel)  
 plt.title(title)  
 plt.show()  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 analyze = Analyze()  
 analyze.create\_neo4j\_node(**'2376916624'**)  
 analyze.create\_neo4j\_relation()  
 analyze.draw\_pie([**'A'**, **'B'**], [2, 3], [0.05, 0], **'test'**)  
 analyze.analyz\_follow\_sex\_ratio()  
 analyze.analyz\_fans\_sex\_ratio()  
 analyze.analyz\_fans\_urank\_distribution()  
 analyze.analyz\_fans\_mbrank\_distribution()  
 analyze.analyz\_follows\_urank\_distribution()  
 analyze.analyz\_follows\_mbrank\_distribution()  
 analyze.analyz\_fans\_verified\_ratio()  
 analyze.analyz\_follows\_verified\_ratio()  
 analyze.analyz\_weibo\_source()  
 analyze.analyz\_weibo\_post\_tendency()

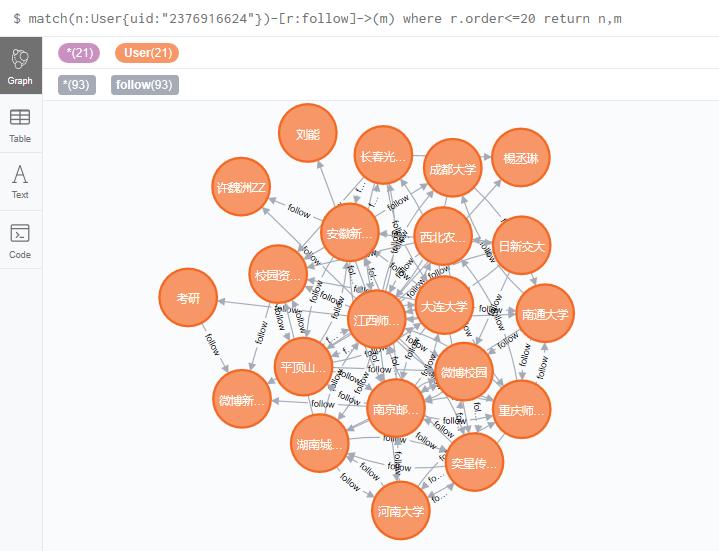
### 结果分析

#### 5.1 用户的社交关系分析

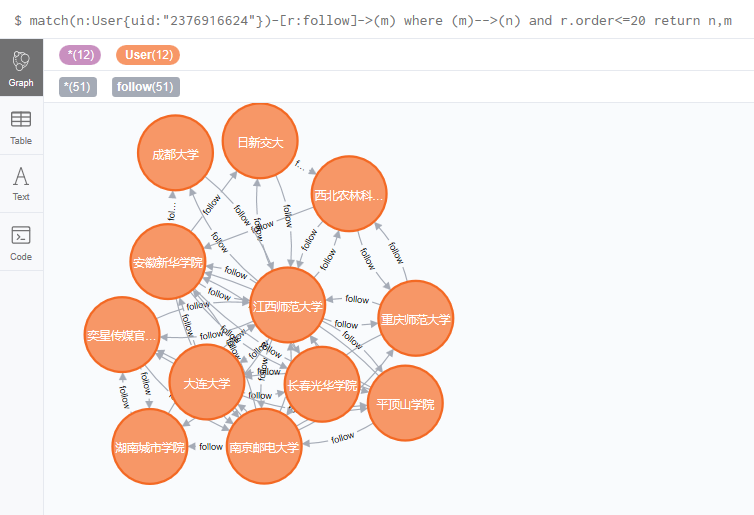
通过neo4j对江西师范大学官方微博的一级关注用户及这些用户之间的相互关系构建出网络图之后发现，一级关注用户仅有193人，但他们之间却形成了4051重关系，可见其关注用户相关性非常高。



通过cypher语句进一步查找其最近关注的20个用户的之间关系网络图发现仅20个用户便构成了93条关系。



通过cypher语句筛选出最近与江西师范大学有相互关注的用户后发现这些用户都是xx大学和xx学院的官方微博。这也就解释了为什么会有如此复杂的用户关系，由此可见大学官方微博账号之间相互关注非常常见。

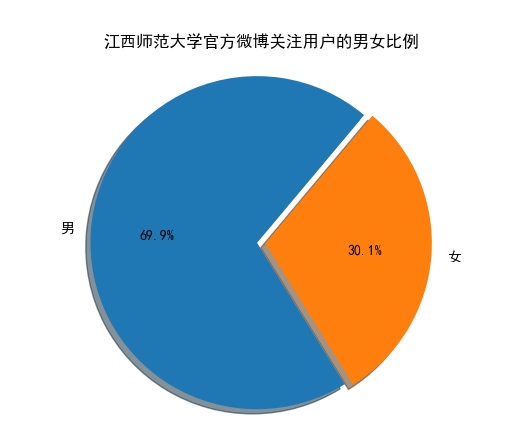


我们还可以进一步找出所有与江西师范大学有相互关注的用户



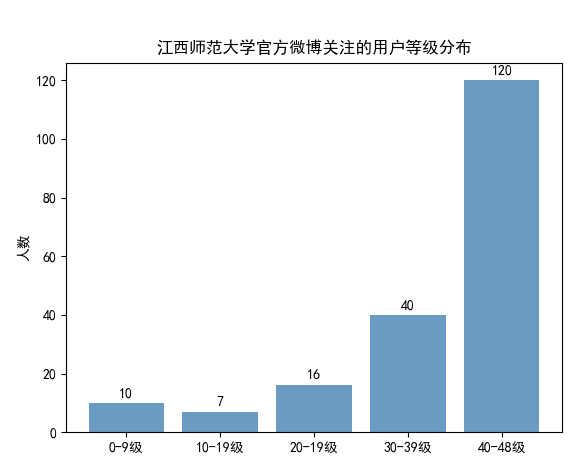
通过查找我们可以知道193名关注的用户中有65名用户与江西师范大学微博账号有相互关注，说明江西师范大学的影响力还可以，有近1/3的关注用户有相互关注。

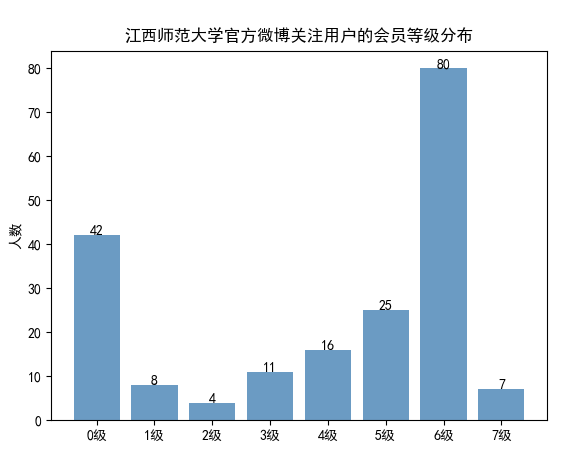
我们还可以继续分析这些关注用户的男女比例信息



通过matplotlib我们可以看出江西师范大学微博的关注用户中男性占了大部分，具体原因我觉得是大多数大学官方认证微博默认为男性。

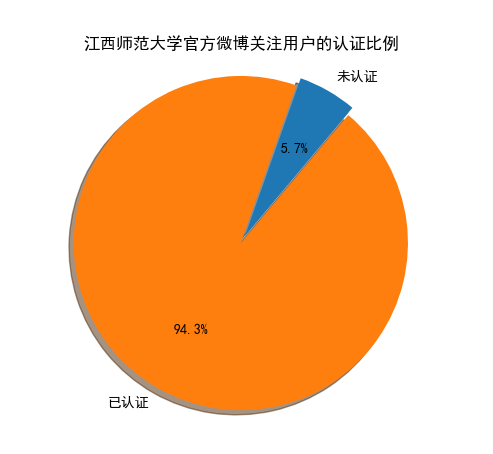
接着继续对关注用户的用户等级及会员等级进行分析





从图中我们可以发现这些用户等级高的占大部分，这进一步验证了江西师范大学微博关注的用户主要是大学的认证机构。

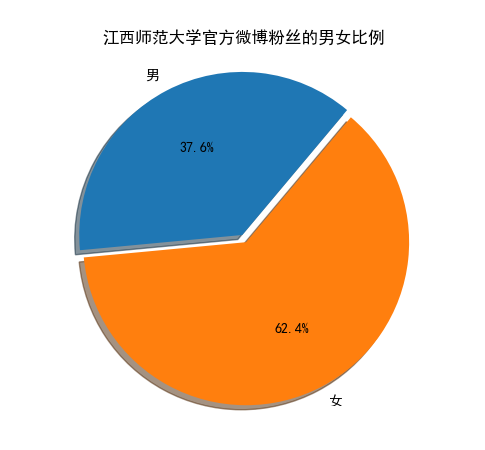
接着继续分析这些关注用户的认证状态



通过图表发现这些用户中认证的比例竟然达到的94.3%，这说明江西师范大学的官方微博很少关注个人用户。

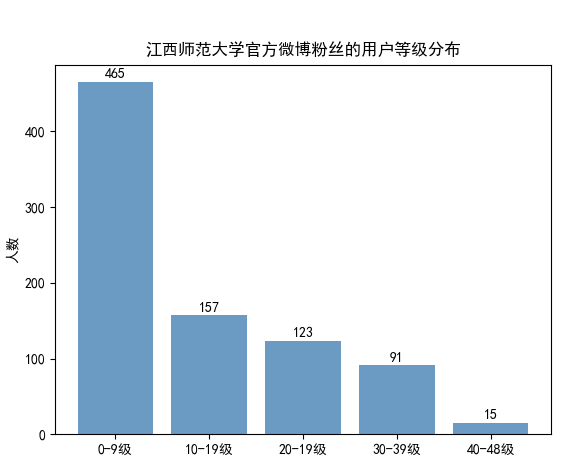
#### 5.2 对粉丝群体的分析

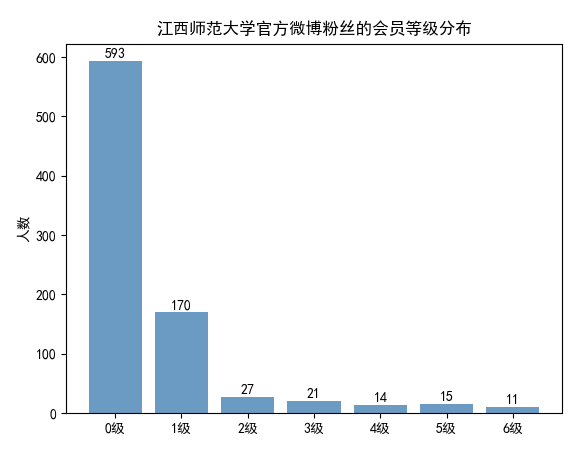
对抓取的2700名左右的粉丝的性别比例进行分析



与关注列表的情况恰恰相反，由于师范类院校女生偏多，江西师范大学的粉丝群体主要为学生，这个比例也从一定程度上反应了学校的男女比例。

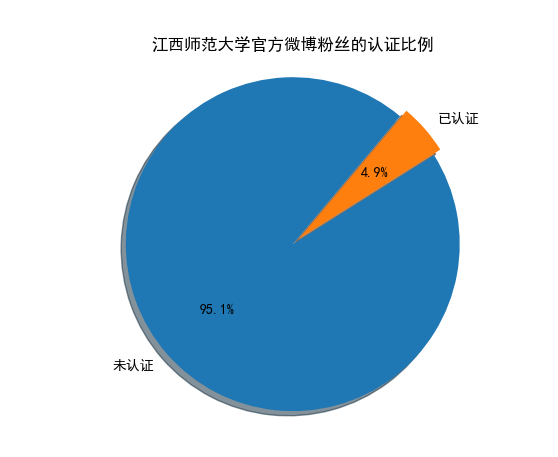
接着，再对粉丝的用户等级和会员等级进行分析





同样与关注用户列表相反，而且高等级的用户只占了很少一部分，还有就是觉大部分的粉丝都没有开通过微博会员，这也从一定程度上反应了师大学生勤俭节约。

同样对粉丝的认证情况进行分析

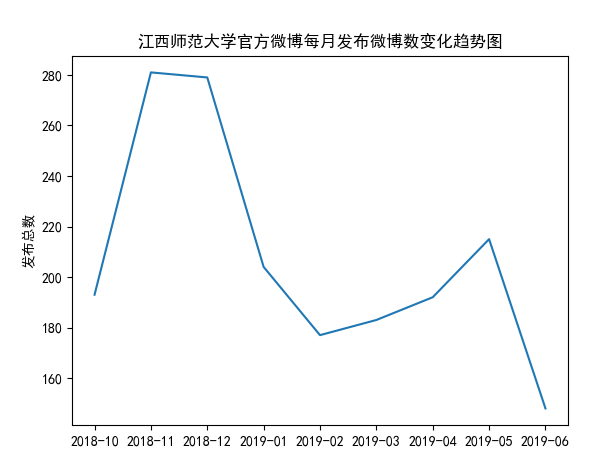


不出所料这些粉丝中认证的比例只占了非常小一步份。

由此我们可以总结出江西师范大学官方微博的关注用户主要是各个大学的认证机构，而粉丝群体主要是学生。

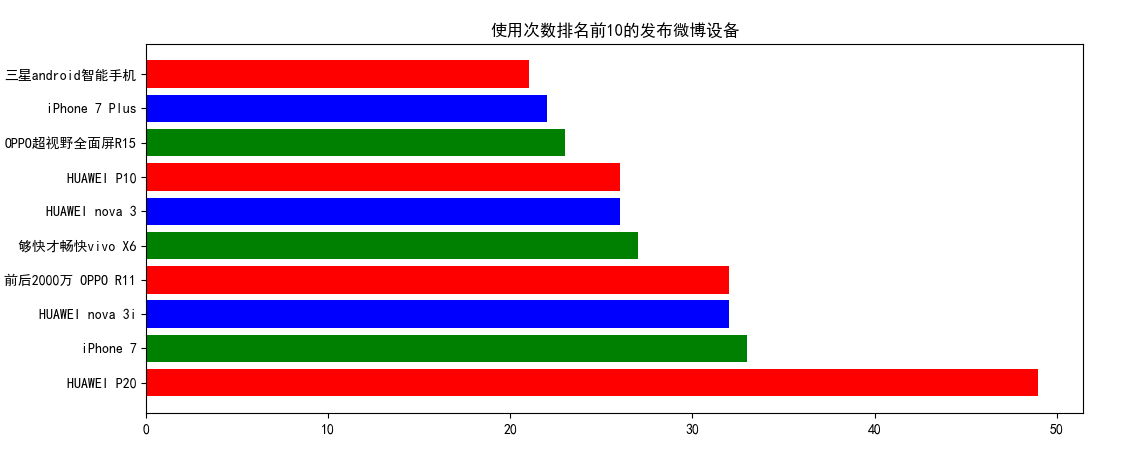
#### 5.3 对发布的微博进行分析

对江西师范大学微博每月的微博发布量进行分析



从图中我们可以看出10，11，12月的微博发布量较1，2，3月多很多，而3，4，5月份开始回涨，到了6月份又开始下跌。据此我们可以猜测12月下跌可能是考试周的影响，因为满意考试工作所以发的微博比较少，而1，2，3月有恰好是假期。根据这个推测我们发现5，6月也符合这个规律，据此我们可以大概知道学校的考试周和假期大概在什么时候。

接着对这些微博的发布来源进行分析，列出10个发布次数最多的设备



从图中我们可以看出除了发布最多的那个设备外，其他设备使用的比例比较均匀，我们可以猜测江西师范大学官方微博的背后运营是有多个用户轮流进行管理的。

### 总结

通过这次python网络编程期末作业的完成，我深刻体会到了python在网络爬虫与数据分析邻域中的便捷于强大，通过这次作业，使我将课堂上的理论知识转化为了实践。当然，由于技术与时间上的原因做的不是很好很完善，在今后的时间里还需要更加努力的学习！