实验五
社交网络数据获取与分析

王亭午, 无210班, 2012011018

2015年4月30号

1 实验目的

1.1 了解并利用用微博API爬取数据

1.2 对获取的数据进行行初步统计分析

2 实验背景

新浪微博是现在对中文社交网络、短文本分析领域重要的分析对象。对于这些平台的数据获取,一般分为两种方式:第一种是通过正常的浏览网页获取完整的HTML/SHTML数据,再进行网页解析获取所需要的信息。第二种方法是通过 API(application programming interface)的方式获取数据。前者的优点是可以 避免平台的权限限制。但是由于新型的前端架构,使得直接获取数据变得困难, 并且解析网页也是一个复杂的工作,往往要随着网页机构变化而重新编写数据分 析部分。而后者,通过API获取数据一般是平台方允许第三方获取没有冗余的 数据的方法。当然同时也对获取的数据进行监控,并进行权限限制。本次实验就 通过新浪微博提供的API获取数据。

3 实验内容

首先根据新浪微博的引导(http://open.weibo.com/),创建一个属于自己的应用。 创建应用成功后,通过右上角我的应用进入到应用界面,在应用信息、高级信息下填写授权回调页等信息。 下载java的SDK后,按照要求修改src/config.properties文件中的信息。 Client_ID对应App Key,client-SERCRET对应App Secret, redirect_URI对应 授权回调页地址。授权回调页是前一步填写的;App Key/Secret可在应用基本信息找到。

3.1 任务一

利用example/timeline/文件夹中所给的获取public timeline的代码, 获取某新浪微博连续 1 小时以上的public time line微博信息。请统计所获得的微博总数,以及数目随时间的变化。

本题存在一个问题，那就是public time line微博信息的**数量实在太大**。 实际上，有三个因素限制了这个题目的完成。
第一：访问频次限制。每次访问只能get到有限条（count小于200），而每小时访问有限制。
第二：网速限制。即使count是无限的，现有的宿舍、实验室网速也无法跟上public time line的吞吐量。
第三：如上所说，public time line的吞吐量**太大**。我曾经尝试使用最高网速最大下载量，每一秒钟下载200条。通过观察获取微博的Id，可以看到，获取速度仍然是小于微博的产生速度。
因此，这个时候我们获取的信息是无效的。 我们更换思维，获取相对来说吞吐量较小Friends Time line。

关键代码如下，代码在task1.java，命令行参数为2.00RQ5RSCfeutRB68006688-3cP5ONTB 2102384553：

```
private static StatusWapper getMsg(String access_token) {
  Timeline tm = new Timeline(access_token);
  StatusWapper status = null;
  try {
    status = tm.getFriendsTimeline(map);
  } catch (WeiboException e) {
    e.printStackTrace();
  }
  if (status == null) {
    System.out.print("reading error!");
    return null;
  }
  else return status;
}
```

在上面的函数中，我们使用对应的token来调用对应的API，获得一个Status Wapper结构。在这个结构中有我们需要的微博信息。

```
public static void main(String[] args) throws IOException {
  map.put("count", "200");
  /* start the program, setting the start time information*/
  Date startDate = new Date();
  System.out.printf("The record starts at %s\n", startDate.toString
    ());

  long msgNum = 0;  /* record the number of Message in one fetch */
  /* initialize the access */
  StatusWapper status;
  String access_token= args[0];
  boolean timeExhaust = true;
  int timecount = -1;
  while (timeExhaust) {
    Date currentDate = new Date();
    timecount ++;

    int minDiff = (int) ((currentDate.getTime()-startDate.getTime()
    )/(60*1000));
```

```
    if (minDiff >= 60) timeExhaust = false;

    status = getMsg(access_token);
    List<Status> msgList = status.getStatuses();
    msgNum = msgList.size();
    if (map.get("since_id") != null & msgNum > 0) msgNum = msgNum -
     1;
    totalNum = totalNum + msgNum;
    numRecord[timecount] = (int) msgNum;
    System.out.printf("%d Message Fetched at time %s\n", msgNum,
        currentDate.toString());
    if (msgNum>0){
      Status iStatus = msgList.get((int) 0);  /* get the new msg */
      String IDstring = iStatus.getId();       /* get the id
    information */
      map.put("since_id", IDstring);
    }

    /* get ready for the next round */
    System.out.printf("Sleeping !\n");
    System.out.printf("Total %d message! !\n", totalNum);
      try {
      Thread.sleep(4 * 60000);
    } catch (InterruptedException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    System.out.printf("Awaking !\n");

  }
  FileOutputStream out = new FileOutputStream(new File("/home/wtw/
    network/task1"));
  out.write(("The overal number of status is " + totalNum + "\n").
    getBytes());
  for (int output = 0; output < 22; output++){
    out.write(("Get number of status is " + numRecord[output] + "\n
    ").getBytes());
  }
  out.close();
}
```

在每次循环的开始，我们通过$(int)((currentDate.getTime()-startDate.getTime())/(60*1000))$;来检查是否超过1个小时，超过的话停止循环。
同时，我们使用一个HashMap来调用API，其中设置$map.put("since_id", IDstring)$;这样就可以做到剔除重复的信息。
在程序的尾端，每次循环后我们都会休眠4分钟，使用$Thread.sleep(4*60000)$;函数来实现。 得到的效果图如下：

```
The overal number of status is 161
Get number of status is 101
Get number of status is 4
Get number of status is 4
Get number of status is 7
Get number of status is 6
Get number of status is 2
```

```
Get number of status is 5
Get number of status is 4
Get number of status is 1
Get number of status is 6
Get number of status is 2
Get number of status is 7
Get number of status is 0
Get number of status is 4
Get number of status is 5
Get number of status is 3
```

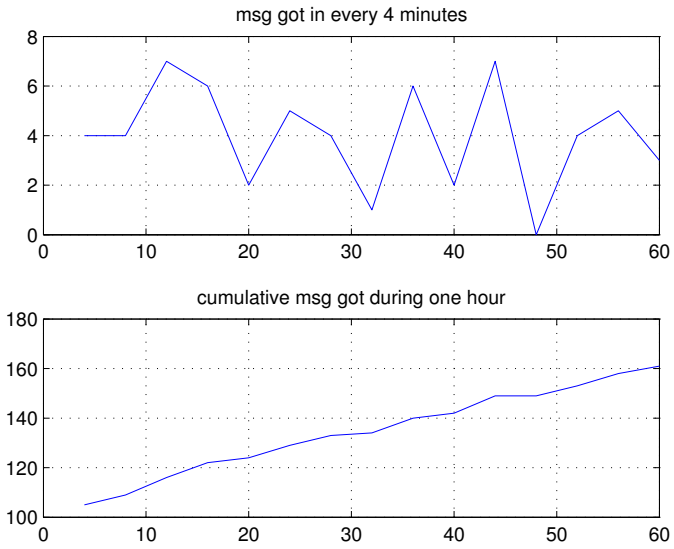


Figure 1: 获取到的关注好友微博数量

可见虽然每4分钟我收到的关注好友的状态数是非常杂乱的，但是累积起来还是基本呈一个漂亮的线性。 在第一次还可以收取到之前的101条消息，我把它作为基准。测量的时间是下午2点到3点。

3.2 任务二

请获取自己的微博信息,包括粉丝、关注列表和所有微博内容,并 统计微博信息以及其粉丝/关注的用户基本信息。包括但不限于:用户发表微博 的频率, 用户的关注/粉丝用户的男女比例,地理分布情况等信息。

注：任务二因为我自己不怎么玩微博（只有9个状态和18个好友），所以在分析粉丝和关注的信息上面借用了陈琳元同学的账号进行分析。 **虽然借用了账号，但是代码完全是独立完成的，两人没有任何关于代码的沟通**。助教老师

可以比对。
在task2.java文件中，我获取了我自己的账号的个人信息，包括粉丝数，关注数量，以及最近的9条微博的发布时间。 输出结果如下

```
The basic information for the user 1612900094 is given as
The user has 18 fans
The user has 179 friends
And the user poorly only publised 9 status

On the 849 th day, the user publised 1 message
On the 534 th day, the user publised 1 message
On the 801 th day, the user publised 1 message
On the 355 th day, the user publised 1 message
On the 312 th day, the user publised 1 message
On the 738 th day, the user publised 1 message
On the 526 th day, the user publised 1 message
On the 573 th day, the user publised 1 message
On the 809 th day, the user publised 1 message
```

其中日期是相对于2013年1月1号00:00的时间。顺序是乱的是因为使用了HASH表。值得注意的是，在最新的API接口文档中说明，此API更新后只能提供最新的5条状态。 我通过使用设置Max_Id,使得能够获取9条状态。我用了我，陈琳元，冯杰三个人的账号进行测试，确认9条是这个API获取的权限极限。 频率可以使用时间间隔的倒数来求取，不过因为我发表的状态太少物理意义不大，这里就不写出来了。 可以看到我有179个关注和18个粉丝。 下面是task2.java的代码。

```
public class task2 {

private static String access_token;
private static String uids;
private static UserCounts myInfo;

public static void main(String[] args) throws IOException,
    ParseException, WeiboException {

  access_token = args[0]; /* it is the access token of me*/
  uids = args[1];

  Users um = new Users(access_token);
  /* get my basic information */
  try {
    List<UserCounts> user = um.getUserCount(uids); /* it is a list
    of UserCounts */
    myInfo = (UserCounts)user.get(0);              /* we get our
    basic info*/
  } catch (WeiboException e) {
    e.printStackTrace();
  }


  long nMyfans = myInfo.getFollowersCount();
  long nFriends = myInfo.getFriendsCount();
  long nStatus = myInfo.getStatusesCount();
```

```
  /* write down the information */
  FileOutputStream out = new FileOutputStream(new File("/home/wtw/
    network/task2"));
  FileOutputStream StatusNum = new FileOutputStream(new File("/home
    /wtw/network/task2_Status"));
  out.write(("The basic information for the user " + uids + " is
    given as\n").getBytes());
  out.write(("The user has " + Long.toString(nMyfans) + " fans\n").
    getBytes());
  out.write(("The user has " + Long.toString(nFriends) + " friends\
    n").getBytes());
  out.write(("And the user poorly only publised " + Long.toString(
    nStatus) + " status\n").getBytes());

  /* the specific information is given */
  out.write(("And the user poorly only publised " + Long.toString(
    nStatus) + " status\n").getBytes());

  System.out.printf("Sorting the information\n");

  DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss
    ");
  Date StartDay = dateFormat.parse("2014-01-01 00:00:00");

  /* get my frequency and one of my friend frequency */
  Map<Integer, Integer> list = getWeiboTime(access_token, uids,
    StartDay);

  for (int key : list.keySet()) {
    System.out.println("key= "+ key + " and value= " + list.get(key
    ));
    StatusNum.write((Integer.toString(key) +"\n").getBytes());
    StatusNum.write((Integer.toString(list.get(key)) +"\n").
    getBytes());
    out.write(("On the " + Integer.toString(key) + " th day, the
    user publised").getBytes());
    out.write((Integer.toString(list.get(key)) + " message\n").
    getBytes());
  }

  /* get my follower and friends location, gender */
  //int mail = getPosition(access_token, uids, StartDay);
  StatusNum.close();
  out.close();
}
```

获取部分和任务一区别不大，主要也是通过设定一个since_Id来筛选信息。
在task3.java文件中，我获取了陈琳元账号的粉丝信息，包括粉丝的性别，地理位置，粉丝他们的粉丝和关注数量。参数为2.00RQ5RSCfeutRB680066883cP5-ONTB 2102384553，代码如下：（task4.java类似，不贴出来了）

```
public static void main(String[] args) throws IOException {
  String access_token = args[0];
  Friendships fm = new Friendships(access_token);
```

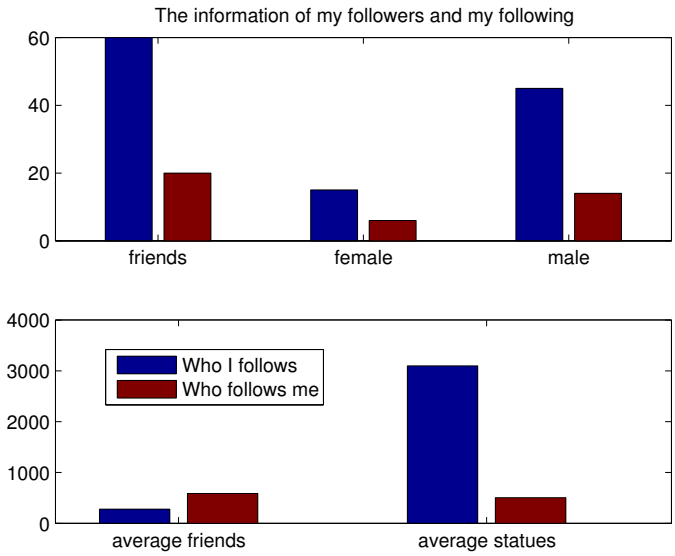
```
String screen_name = args[1];

/* use a list to record the */
Map<Integer, Integer> pro2User= new HashMap<Integer, Integer>();
Map<Integer, String> pro2Str= new HashMap<Integer, String>();
int femaleNum = 0;
int maleNum = 0;
int friendsNum = 0;
int followNum= 0;
int statuesNum = 0;
int AverageFollowing = 0;
int AverageFollowMe = 0;

Map<String, String> map = new HashMap<String, String>();  /* it
  is the parameter map */
map.put("count", "200");
map.put("screen_name", screen_name);

try {
  UserWapper users = fm.getFollowers(map);
  //UserWapper users = fm.getFriends(map);
  for(User u : users.getUsers()){
    /* get the city information */
    int proCode = u.getProvince();

    if (pro2User.get(proCode) != null) {  /* it is old one*/
      pro2User.put(proCode, pro2User.get(proCode) + 1); /* add
  one to this city */
    } else {
      pro2User.put(proCode, 1); /* add one to this city */
      String CityName = u.getLocation().substring(0, 2);
      System.out.print(CityName);
      pro2Str.put(proCode, CityName);
    }

    /* get the gender information */
    if (u.getGender().equals("f")) femaleNum = femaleNum + 1;
    else if (u.getGender().equals("m")) maleNum = maleNum + 1;

    /* get the Friend info */
    friendsNum += u.getFriendsCount();
    followNum += u.getFollowersCount();
    statuesNum += u.getStatusesCount();

    /* follow me */
    if (u.isFollowMe()) AverageFollowMe += 1;
    if (u.isFollowing()) AverageFollowing += 1;

  }
  FileOutputStream out = new FileOutputStream(new File("/home/wtw
  /network/task3"));
  float userNum = (float)users.getUsers().size();
  out.write(("The number of all friends is " + userNum + "\n").
  getBytes());
  out.write(("The number of all female is " + femaleNum + "\n").
  getBytes());
  out.write(("The number of all male is " + maleNum + "\n").
```

```
    getBytes());
    out.write(("The average friends is " + friendsNum / userNum + "
    \n").getBytes());
    out.write(("The average follow is " + followNum / userNum + "\n
    ").getBytes());
    out.write(("The average statues number is " + statuesNum /
    userNum + "\n").getBytes());
    out.write(("The average follow me number is " + AverageFollowMe
     / userNum + "\n").getBytes());
    out.write(("The average following number is " +
    AverageFollowing / userNum + "\n").getBytes());

    for (int key : pro2User.keySet()) {
      out.write(("The province of " + pro2Str.get(key) + " has " +
    pro2User.get(key) + "\n").getBytes());
    }
    out.close();
  } catch (WeiboException e) {
    e.printStackTrace();
  }
}
```

在上面的代码中有一个小技巧，就是使用两个list来分别记录某个城市的人数，和这个城市对应的名字。这样可以节省存储空间，同时也省去了我们插表的时间。获得的结果如下，可以看到，无论是陈琳元关注的和关注他的人，最多的都是来自北京的。在陈琳元关注的人里面，有很多人是没有填写所在地而被归类为other的，我们可以猜测很多公众号都会选择不填这一项来获得更加多的关注。而关注陈琳元的人中，来自广东和海外的数量明显增加，我们可以猜测陈老板一定是个贵人，身边都是一些达官显赫。 粉丝的其他信息如下：

```
The number of all friends is 20.0
The number of all female is 6
The number of all male is 14
The average friends is 587.95
The average follow is 169.15
The average statues number is 504.6
The average following number is 0.45

The number of all friends is 60.0
The number of all female is 15
The number of all male is 45
The average friends is 277.66666
The average follow is 2301973.2
The average statues number is 3096.4
The average follow me number is 0.13333334
```

从上面可以看出，陈琳元关注的对象中，只有13.3%是关注陈琳元的，但是关注陈琳元的人中，高达45%同是也是陈琳元关注的对象。 可见陈琳元关注了不少公众号和大V。
这一点也可反应在发表状态数上，陈琳元关注的人中，平均发表状态数高达3096.4，是关注他的人的6倍(504.6)。

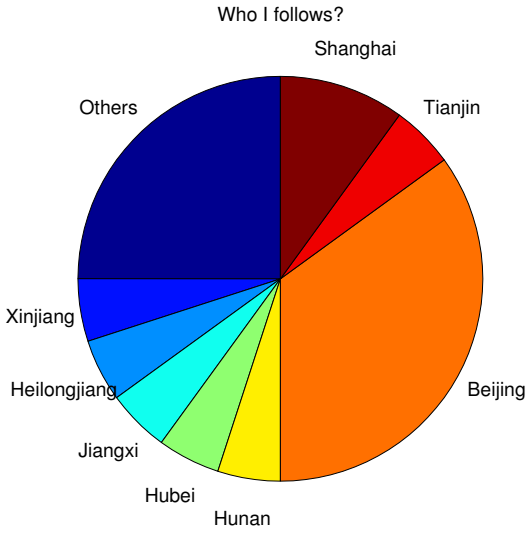


陈琳元关注的人，平均拥有粉丝2301973.2个，远高于关注他的人的粉丝数169.15个。不过两者在关注数量上倒是差不多。
关注的人中，男女比为75%:25%，粉丝中男女比为70%:30%，比例接近。 最后我们画出陈琳元关注和好友的地区分部，两个饼状图如下：

4 思考题

使用API获取数据与爬取网页有何区别?优劣势如何?
答案：API是官方提供的一些接口，你可以用这些接口获得对应标准化格式的信息，用来进行处理。 而爬取网页则是基于我们的HTTP协议，取解析并且根据结果进一步发送对应协议从而获得协议。 从官方的角度看，爬取网页和一个普通用户的行为是一致的，而API就和用户差别很大了。
使用API最大的好处就是简单方便，不用担心自己在管理者眼中是一个需要提防的对象。 但是坏处也非常多，最明显的就是功能太少太差（新浪微博的API尤其明显）。 因为是对方提供的API,很多功能是没有的。比如我是无法获取其他人的微博信息的（在老版API是可以的，更新后反而不行了）。 爬取网页自由度就很大，理论上你可以爬到任何可能的数据。但是工作量比较大，写对应代码时间比较长。


Who follow Me?
Others
overseas
Fujian
Sichuan
Jiangsu
Qinghai
Hunan
Guangdong
Shanghai
Tianjin
Beijing


Who I follows?
Shanghai
Others
Tianjin
Xinjiang
Heilongjiang
Beijing
Jiangxi
Hubei
Hunan