Archive Album

使用Python分析社交网络数据 AUG,03 2014

在线社交网站为人们提供了一个构建社会关系网络和互动的平台。每一个人和组织都可以通过社交网站互动、获取信息并发出自己的声音，因而吸引了众多的使用者。作为一个复杂的社会系统，在线社交网站真实地记录了社会网络的增长以及人类传播行为演化。通过抓取并分析在线社交网站的数据，研究者可以迅速地把握人类社交网络行为背后所隐藏的规律、机制乃至一般性的法则。

然而在线社交网络数据的获取方法有别于线下社会数据的获取（如普查、社会调查、实验、内容分析等）、数据的规模往往非常大（称之为"大数据"并不为过）、跨越的时间范围也相对较长（与社会调查中的横截面数据相比），常规的数据分析方法并不完全适用。例如传统的社会调查的数据往往样本量有限，而在线社交网络中的样本量可以达到千万甚至更多。因而，研究者迫切得需要寻找新的数据获取、预处理和分析的方法。本章的内容具体包括数据的抓取、数据预处理、数据可视化和数据分析部分。

Python简介

本章将简要介绍使用python分析社交网络数据的方法。Python是一种广泛使用的高级编程语言，具有可读性强、编写容易、类库丰富等特点。作为一种"胶水语言"，它可以将使用其他语言编写的各种模块（尤其是C/C++）轻松地联结在一起。自从1991年推出第一个正式版本，因其使用方便，Python社区迅速发展，越来越多的程序员开始使用Python编写程序并贡献了各种功能强大的类库。它被TIOBE编程语言排行榜评为"2010年度编程语言"。

除了免费、功能强大、使用者众多之外，与R和MATLAB相比，Python是一门更易学、更严谨的程序设计语言。如同其它编程语言一样，Python语言的基础知识包括：类型、列表（list）和元组（tuple）、字典（dictionary）、条件、循环、异常处理等。关于这些，初阶读者可以阅读《Beginning Python》一书（Hetland, 2005)。作为一个相对非常完善的编程语言，使用Python编写的脚本更易于理解和维护。

另外，Python中包含了丰富的类库。众多开源的科学计算软件包都提供了Python的调用接口，例如著名的计算机视觉库OpenCV。Python本身的科学计算类库发展也十分完善，例如NumPy、SciPy和matplotlib等。就社会网络分析而言，igraph, networkx, graph-tool, Snap.py等类库提供了丰富的网络分析工具。

读者可以根据个人电脑的操作系统安装相应的Python版本。目前最新的Python版本为3.0，但是通常使用者会选择使用更稳定的2.7版本。虽然使用者也可以使用文本编辑器编写代码，但是使用体验不如使用好的编译器。编译器是编写程序的重要工具。目前，免费的Python编译器

有Spyder、PyCharm(免费社区版)、Ipython、Vim、 Emacs、 Eclipse(加上PyDev插件)。对于使用Windows操作系统的用户，推荐使用Winpython。Winpython内置了Spyder为编译器，与Python(x,y)相比大小适中；免安装，下载后解压即可用；安装类库很方便，并且内置了NumPy、SciPy等类库。

数据抓取

目前社交网站的公开数据很多，为研究者检验自己的理论模型提供了很多便利。例如斯坦福的社会网络分析项目就分享了很多相关的数据集。社交网站为了自身的发展，往往也通过各种合作项目（例如腾讯的"犀牛鸟项目"）和竞赛（例如Facebook通过Kaggle竞赛公布部分数据）向研究者分享数据。

但是，有时候研究者还是被迫需要自己收集数据。受限于网站本身对于信息的保护和研究者自身的编程水平，互联网数据的抓取过程依然存在众多问题。以下，我们将从三个方面着手简要介绍使用Python进行数据抓取的问题：直接抓取数据、模拟登录抓取数据、基于API接口抓取数据。

一、直接抓取数据

通常的数据抓取遵循可见即可得的规律，即可以观察到的，就可以被抓取。对于网页内容的抓取，可以是把整个网页都存下来，回头再清洗。这样做比较简单有效，但是还是回避不了之后的从html文件中进行的数据提取工作。在下面的例子当中，我们将尝试抓取百度新闻页面（http://news.baidu.com/）的热点新闻。在这个例子当中，我们要使用urllib2这个类库来获取该网页的html文本。

在获取html之后，我们将使用一个流行的类库BeautifulSoup来解析html并提取我们需要的信息。现在的BeautifulSoup已经发展到第四个版本。可以使用easy_install或者pip install的方法安装。如果读者使用的是Spyder的话，可以点击Tools--Open command prompt。然后，在打开的命令窗口中输入：easy_install beautifulsoup4 就可以了。

```
easy_install beautifulsoup4
```

使用beautifulsoup解析中文html的时候遇到的主要问题多是由encoding造成的。需要使用sys设定默认的encoding方式为gbk，并在BeautifulSoup函数中指定from_encoding为gbk。

```
import urllib2
from bs4 import BeautifulSoup

#设置默认encoding方式
import sys
reload(sys)
sys.setdefaultencoding('gbk')

url = 'http://news.baidu.com/' #待抓取的网页地址
content = urllib2.urlopen(url).read() #获取网页的html文本
#使用BeautifulSoup解析html
soup = BeautifulSoup(content, from_encoding = 'gbk')
#识别热点新闻
hotNews = soup.find_all('div', {'class', 'hotnews'})[0].find_all('li')
for i in hotNews:
    print i.a.text #打印新闻标题
    print i.a['href'] #打印新闻链接
```

这样就可以抓取当天的热点新闻，输出的结果如下：

```
习近平：改革惟其艰难 才更显勇毅
http://china.cankaoxiaoxi.com/2014/0808/454139.shtml
治疗费政府全担
```

```
http://gb.cri.cn/42071/2014/08/06/2165s4643613.htm
李克强点赞人物盘点：有女县长也有棒棒军
http://news.youth.cn/jsxw/201408/t20140808_5605703.htm
李克强指示地震救灾
http://yn.yunnan.cn/html/2014-08/07/content_3316250.htm
云南地震致615人遇难
http://news.baidu.com/z/ynlddz/new/zhuanti.html
临时安置点1顶帐篷内住20人
http://news.youth.cn/gn/201408/t20140808_5607225.htm
```

二、模拟浏览器抓取数据

越来越多的网站要求必须登录才能看到内容，这个时候就需要使用编程软件模拟浏览器登录。登录成功后，就可以抓取内容了。这里举一个抓取聊天论坛帖子列表的例子。这个网站的网络链接为：http://members.lovingfromadistance.com/forum.php， 我们首先写一个叫screen_login的函数。其核心是定义个浏览器对象br = mechanize.Browser()。这个时候，需要借用浏览器的cookie功能，主要借助于cookielib包。代码如下所示：

```
import mechanize
import cookielib

def screen_login():
    br = mechanize.Browser()
    cj = cookielib.LWPCookieJar()
    br.set_cookiejar(cj)
    # setting
    br.set_handle_equiv(True)
    br.set_handle_redirect(True)
    br.set_handle_referer(True)
    br.set_handle_robots(False)
    # br.set_debug_http(True)
    # User-Agent (this is cheating, ok?)
    br.addheaders = [('User-agent', 'Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.0.1) Gecko/2008071615 Chrome/17.0.963.56 Fedora/3
.0.1-1.fc9 Firefox/3.0.1')]
    br.set_handle_refresh(mechanize._http.HTTPRefreshProcessor(), max_time=1)
    # Open the login page
    br.open('http://members.lovingfromadistance.com/login.php?do=login')
    br.select_form(nr = 0) # Find the login form
    br['vb_login_username'] = '你的用户名'
    br['vb_login_password'] = '你的注册密码'
    br.submit() # Submit the form
    return(br)

br = screen_login()
```

为了从HTML文档提取cookies，首先使用cookielib模块的LWPCookieJar()函数创建一个cookie jar的实例。LWPCookieJar()函数将返回一个对象，该对象可以从硬盘加载Cookie，同时还能向硬盘存放Cookie。之后，通过 br.set_cookiejar(cj)将这个cookie jar关联到mechanize的浏览器对象br上。简单设置一些浏览器属性后，需要定义使用的user-agent。用户代理（User Agent）指的是代表使用者行为的软件，主要是设置浏览器的头文件。

最后是关键的一步，打开登录页面，输入用户名和用户密码。需要使用br.select_form(nr = 0)来找到登录表格。这里nr的设置比较灵活，不同网站的数值不同。然后输入用户名和密码。比如：br['vb_login_username'] = 'Your registered User name'， 这里的vb_login_username也会随着网站本身使用的具体内容而不同。运行br = screen_login()就可以模拟登录成功，然后就可以开始数据抓取和使用BeautifulSoup来进

行信息提取的工作了，此处不再赘述。

三、基于**API**接口抓取数据

好在随着数字化媒体浪潮的到来，第三方开发的网站应用已经成为社交网络必不可少的一部分。社交网站为了自身的发展，往往选择向外界开放部分资源，以方便第三方发展基于该社交网站的产品，进而更好吸引使用者使用。比如新浪微博上有着各种不同的APP，这些应用的数据接口（API）就是由新浪微博所提供的。

不同的编程语言与通用的API接口之间是由软件开发工具包SDK(Software Development Kit）衔接在一起的。仍然以新浪微博的SDK为例，打开其页面（http://open.weibo.com/wiki/SDK）我们会发现对应于各种编程语言的SDK，有些由新浪微博官方提供的，有些则是由广大使用者编写的。就Python而言，新浪微博官方推荐的Python SDK是sinaweibopy。sinaweibopy是纯Python编写的单个文件，代码简洁，无依赖，运行可靠。

安装sinaweibopy的方法非常简单，只需要打开的命令窗口中输入：easy_install sinaweibopy 就可以了（https://pypi.python.org/pypi/sinaweibopy/1.1.3）。

```
easy_install sinaweibopy
```

数据抓取的第一步，就是建立数据连接的工作，以获取社交网站开放数据流的许可。当然，这首先需要使用者注册一个app。以新浪微博为例，研究者可到其应用开发页面注册 。这样，使用者可以获取一个APP_KEY和对应的APP_SECRET。

现在流行的方式是使用OAuth获取连接社会化媒体的API的使用权限。它工作的原理非常简单：1.首先使用者发出使用请求，2.然后新浪微博在收到请求后向使用者发出一个授权码，3.获取授权码之后使用者依据授权码从新浪微博获取连接码（ACCESS TOKEN），4.使用连接码，使用者就可以连接到新浪微博的数据库并获取数据了。以上过程可以使用以下Python代码来实现：

```
from weibo import APIClient
import urllib2
import urllib
import sys

def weiboClient():
    APP_KEY = '663049101' # app key
    APP_SECRET = '2fc9ed9a3b9e7f37c3e6667464f0617e' # app secret
    CALLBACK_URL = 'https://api.weibo.com/oauth2/default.html' # callback url
    AUTH_URL = 'https://api.weibo.com/oauth2/authorize'
    USERID = 'w4'
    PASSWD = 'w' #your pw
    client = APIClient(app_key=APP_KEY, app_secret=APP_SECRET, redirect_uri=CALLBACK_URL)
    referer_url = client.get_authorize_url()
    print "referer url is : %s" % referer_url
    cookies = urllib2.HTTPCookieProcessor()
    opener = urllib2.build_opener(cookies)
    urllib2.install_opener(opener)
    postdata = {"client_id": APP_KEY,
            "redirect_uri": CALLBACK_URL,
            "userId": USERID,
            "passwd": PASSWD,
            "isLoginSina": "0",
            "action": "submit",
            "response_type": "code",
               }
    headers = {"User-Agent": "Mozilla/5.0 (Windows NT 6.1; rv:11.0) Gecko/20100101 Firefox/11.0",
               "Host": "api.weibo.com",
```

```
            "Referer": referer_url
            }
    req  = urllib2.Request(
       url = AUTH_URL,
       data = urllib.urlencode(postdata),
       headers = headers
       )
    try:
        resp = urllib2.urlopen(req)
        print "callback url is : %s" % resp.geturl()
        code = resp.geturl()[-32:]
        print "code is : %s" %  code
    except Exception, e:
        print e
    r = client.request_access_token(code)
    access_token1 = r.access_token # The token return by sina
    expires_in = r.expires_in

    print "access_token=" ,access_token1
    print "expires_in=" ,expires_in   # access_token lifetime by second.
    client.set_access_token(access_token1, expires_in)
    return client
```

在上述代码中，我们定义了一个名为weiboClient的函数。完成以上步骤之后，使用者只需要运行client = weiboClient()的程序，就可以连接到新浪微博的API接口了。

下一步是通过查阅社交网站的API文档，选取适当的API接口，就可以很方便地从社交网站抓取数据了。因为直接从网站数据库获取数据，因而数据结构化较好。获取数据使用许可之后，其使用就非常方便灵活了。2014年8月，云南鲁甸县发生大地震，人民日报官方微博8月7日19:51来自人民日报微博 |举报发微博报道了最新的死亡人数。消息内容称：

人民日报：#云南鲁甸县地震#【遇难人数增至615人[蜡烛]】据抗震救灾指挥部消息，截至今天19时，地震共造成615人死亡，其中鲁甸县526人、巧家县76人、昭阳区1人、会泽县12人；目前仍有114人失踪，其中鲁甸县109人、巧家县5人；另有3143人受伤。

这里需要注意的是每一条微博的号码有两种表示形式：一种是字母和数字的组合，另一种是数字。由该条微博的网络链接，我们可以得到前者为'Bhd8k0Jv8'。这个时候，我们可以通过statuses__queryid这个API接口可以将它转化为纯数字的形式。其它更多的新浪微博API接口可以参阅http://open.weibo.com/wiki/API%E6%96%87%E6%A1%A3_V2。

```
mid = client.get.statuses__queryid(mid = 'Bhd8k0Jv8', isBase62 = 1, type = 1)['id']
```

这里，我们想要看一下这条微博的转发网络，并计算其网络特征。将主要用到的是API接口中的statuses__repost_timeline。在这个数据抓取过程中，每次可以抓取一个页面上的200条转发信息，根据总转发量，我们可以计算需要抓取的页面总数。由此，我们需要先定义一个函数，以确定转发页面的数量。如以下代码所示：

```
#定义确定转发页面数量的函数
def getPageNum(mid):
    count = client.get.statuses__count(ids = mid)
    repostNum = count[0]['reposts']
    if repostNum%200 == 0:
        pages = repostNum/200
    else:
        pages = int(repostNum/200) + 1
    return pages
```

```
# 定义抓取转发的函数
def getReposts(mid, page):
    r = client.get.statuses__repost_timeline(id = mid, page = page, count = 200)
    if len(r) == 0:
        pass
    else:
        m = int(len(r['reposts'])) # 该页面里的微博转发数量
    try:
        for i in range(0, m): #使用for循环遍历该页面里的所有转发微博
            """1.1 转发微博的属性"""
            mid = r['reposts'][i].id
            text = r['reposts'][i].text.replace(",", "")
            created = r['reposts'][i].created_at
            reposts_count = r['reposts'][i].reposts_count
            comments_count = r['reposts'][i].comments_count
            """1.2 微博转发者的属性"""
            user = r['reposts'][i].user
            user_id = user.id
            user_name = user.name
            user_province = user.province
            user_city = user.city
            user_gender = user.gender
            user_url = user.url
            user_followers = user.followers_count
            user_friends = user.friends_count
            user_statuses = user.statuses_count
            user_created = user.created_at
            user_verified = user.verified
            """2.1 源微博的属性"""
            rts = r['reposts'][i].retweeted_status
            rts_mid = rts.id
            #rts_text = rts.text.replace(",", "")
            rts_created = rts.created_at
            rts_reposts_count = rts.reposts_count
            rts_comments_count = rts.comments_count
            """2.2 源微博发出者的属性"""
            rtsuser_id = rts.user.id
            rtsuser_name = rts.user.name
            rtsuser_province = rts.user.province
            rtsuser_city = rts.user.city
            rtsuser_gender = rts.user.gender
            rtsuser_url = rts.user.url
            rtsuser_followers = rts.user.followers_count
            rtsuser_friends = rts.user.friends_count
            rtsuser_statuses = rts.user.statuses_count
            rtsuser_created = rts.user.created_at
            rtsuser_verified = rts.user.verified
            timePass = clock()-start
            if round(timePass) % 2 == 0:
                print mid, rts_mid, "I have been working for %s seconds" % round(timePass)
                time.sleep( random.randrange(3, 9, 1) )  # To avoid http error 504 gateway time-out
```

```
            print >>dataFile, "%s,'%s','%s',%s,%s,%s,%s,%s,'%s',%s,%s,%s,'%s',%s,%s,%s,'%s',%s,%s,'%s',%s,'%s',%s,%s,%s,'%s',%s,%s,%s
,%s,%s"  % (mid, created, text, reposts_count, comments_count,rts_reposts_count, rts_comments_count, user_id, user_name, user_provinc
e, user_city, user_gender,  user_url, user_followers, user_friends, user_statuses, user_created, user_verified,  rts_mid, rts_created
, rtsuser_id, rtsuser_name, rtsuser_province, rtsuser_city, rtsuser_gender, rtsuser_url, rtsuser_followers, rtsuser_friends, rtsuser_
statuses, rtsuser_created, rtsuser_verified)
    except Exception, e:
        print >> sys.stderr, 'Encountered Exception:', e, page
        time.sleep(120)
        pass
```

定义了以上函数之后，我们可以很容易地抓取并存储数据。代码如下：

```
client = weiboClient() #连接到API接口
#定义存储文档地址
dataFile = open("D:/github/weibo_repost/weibo_repost_all.csv",'wb')
start = clock() #定义初始时间
#使用for循环遍历所有的待抓取页面
for page in range(1, pageNum + 1):
    getReposts(mid, page) # 抓取单页的转发信息
dataFile.close() # 关闭存储文件
```

数据预处理

大多数时候，抓取的数据往往并不能直接满足我们分析的需求，往往还需要对数据进行预处理。对于本章所涉及到的微博转发网络而言，主要是要理解二度转发的过程。如下图所示：

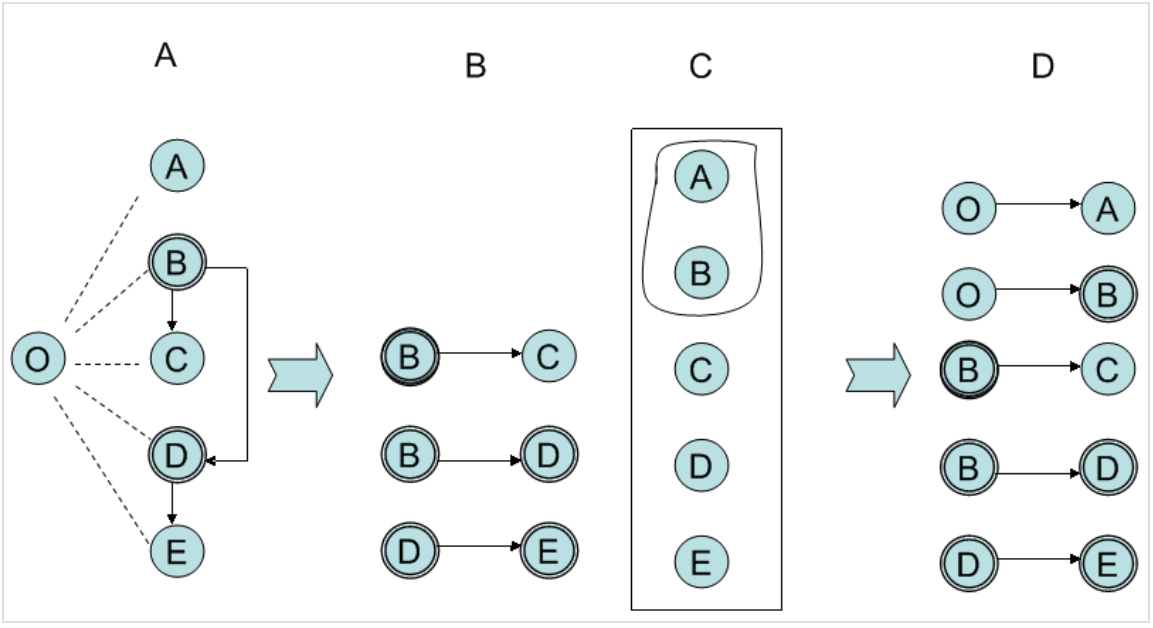


图1：二度转发和微博扩散网络

新浪微博不同与Twitter的一个地方在于，在一条微博的页面中记录了所有转发过这条微博的情况（除非被删除）。如上图1-A中所示，我们用O来标示源微博，五个转发者分别是A、B、C、D、E。我们可以知道就这条消息，五个转发者又被转发的次数。比如，B和D又分别被转发了2次和1次。我们把这种转发称之为二度转发（注意：二度转发这里是一个非常狭隘的定义，它指的是源微博的转发者相互之间转发的情况）。二度转发可以用连边列表（edgelist）的形式表示并存储。在这个连边列表当中，第一列表示信息流出点，第二列表示信息流入点，如上图1-B所示。因此，通过抓取二度转发的情况，我们可以知道信息流入点（这里即C、D、E）都已经确定了其信息来源。只有未出现在二度转发第二列的点（即A、B）的信息来源不在信息转发者当中，那么其信息来源就只能是源微博O，如上图1-C所示，这样我们就可以补全二度转发的连边列表，如上图1-D所示。

基于以上信息，要想获取完整的转发网络，我们需要先获得二度转发网络。

```
#定义二度转发的函数
def get2stepReposts(mid, uid, page):    #将被二度转发的微博的mid和uid传递进来
    try:
        r = client.get.statuses__repost_timeline(id = mid, page = page, count = 200)
        if len(r) == 0:
            pass
        else:
            m = int(len(r['reposts']))
        for i in range(0, m):
            """1.1 转发微博的属性"""
            rt_mid = r['reposts'][i].id
            created = r['reposts'][i].created_at
            """1.2 微博转发者的属性"""
            user = r['reposts'][i].user
            rt_uid = user.id
            print >>dataFile2, "%s,%s,%s,%s,'%s'"  % (mid, uid, rt_mid, rt_uid, created)
    except Exception, e:
        print >> sys.stderr, 'Encountered Exception:', e, page
        time.sleep(120)
        pass
```

定义了抓取二步转发的函数之后，就可以抓取二步转发网络了。

```
with open("D:/github/weibo_repost/weibo_repost_all.csv") as f:
    for line in f:
        line = line.strip().split(',')
        repostCount = int(line[3])
        mid2step = line[0]
        uid = line[7]
        if repostCount > 0:
            print mid2step, repostCount
            if repostCount%200 == 0:
                pages = repostCount/200
            else:
                pages = int(repostCount/200) + 1
            dataFile2 = open("D:/github/weibo_repost/weibo_repost_2_step.csv",'a')
            for page in range(1, pages + 1):
                get2stepReposts(mid2step, uid, page)
            dataFile2.close()
```

在获取了二步转发数据之后，我们首先得到所有的转发者列表，然后获取二步转发网络中的信息流入节点，并对照二者的差异以找出直接从源微博转发的情况。

```
#首先得到所有的转发者列表
with open("D:/github/weibo_repost/weibo_repost_all.csv") as f:
    mids = []
    for line in f:
        line = line.strip().split(',')
        mid = line[0]
```

```
        mids.append(mid)

#然后获取二步转发网络中的信息流入节点
with open("D:/github/weibo_repost/weibo_repost_2_step.csv") as f:
    toNodes = []
    for line in f:
        line = line.strip().split(',')
        mid = line[2]
        toNodes.append(mid)
#找到未出现在二步转发网络中的信息流入节点的转发者
oneStep = []
for i in mids:
    if i not in toNodes:
        oneStep.append(i)
#构建一步转发数据
with open("D:/github/weibo_repost/weibo_repost_all.csv") as f:
    for line in f:
        line = line.strip().split(',')
        created = line[1]
        mid = line[0]
        uid = line[7]
        rtmid = line[18]
        rtuid= line[20]
        if mid in oneStep:
            with open('D:/github/weibo_repost/weibo_repost_1_step.csv', 'a') as g:
                record = rtmid + ','+ rtuid+','+ mid + ','+uid+','+created
                g.write(record+"\n")
#合并一步转发和二步转发，得到完整的转发网络。
with open("D:/github/weibo_repost/weibo_repost_1_step.csv") as f:
    for line in f:
        with open('D:/github/weibo_repost/weibo_reposts_network.csv', 'a') as g:
            g.write(line)
with open("D:/github/weibo_repost/weibo_repost_2_step.csv") as f:
    for line in f:
        with open('D:/github/weibo_repost/weibo_reposts_network.csv', 'a') as g:
            g.write(line)
```

到这里，我们就得到了完整的转发网络。

可视化

为了展现时间的先后顺序，我们首先提取转发网络中的转发时间

```
# 首先，提取转发时间信息
with open('D:/github/weibo_repost/weibo_repost_network.csv', 'r') as f:
    rt_time = []
    for line in f:
        time= line.strip().split(',')[-1]
        day = time[7:9]
        hms= time[9:17].replace(':', '')
        time = int(day + hms)
        rt_time.append(time)
```

```
 #计算转发时间的先后顺序
import numpy
array = numpy.array(rt_time)
order = array.argsort()
ranks = order.argsort()
```

然后我们可以构建转发网络。

```
# 构建微博转发网络
import matplotlib.pyplot as plt
import networkx as nx
G = nx.Graph()
with open('D:/github/weibo_repost/weibo_repost_network.csv', 'r') as f:
    for position, line in enumerate(f):
        mid, uid, rtmid, rtuid= line.strip().split(',')[:-1]
        G.add_edge(uid, rtuid, time = ranks[position])
edges,colors = zip(*nx.get_edge_attributes(G,'time').items())
degree=G.out_degree()#计算节点的出度
path_length = nx.all_pairs_shortest_path_length(G)
depth =[ path_length['2803301701'][i] for i in degree.keys()]
pos=nx.spring_layout(G) #设置网络的布局
fig = plt.figure(figsize=(10, 8),facecolor='white')
nx.draw(G, pos, nodelist = degree.keys(),
        node_size = [np.sqrt(v+1)*10 for v in (degree.values())], node_color = depth,
        node_shape = 'o', cmap=plt.cm.gray,
        edgelist = edges, edge_color = 'gray', width = 0.5,
        with_labels = False, arrows = False)
plt.show()
```

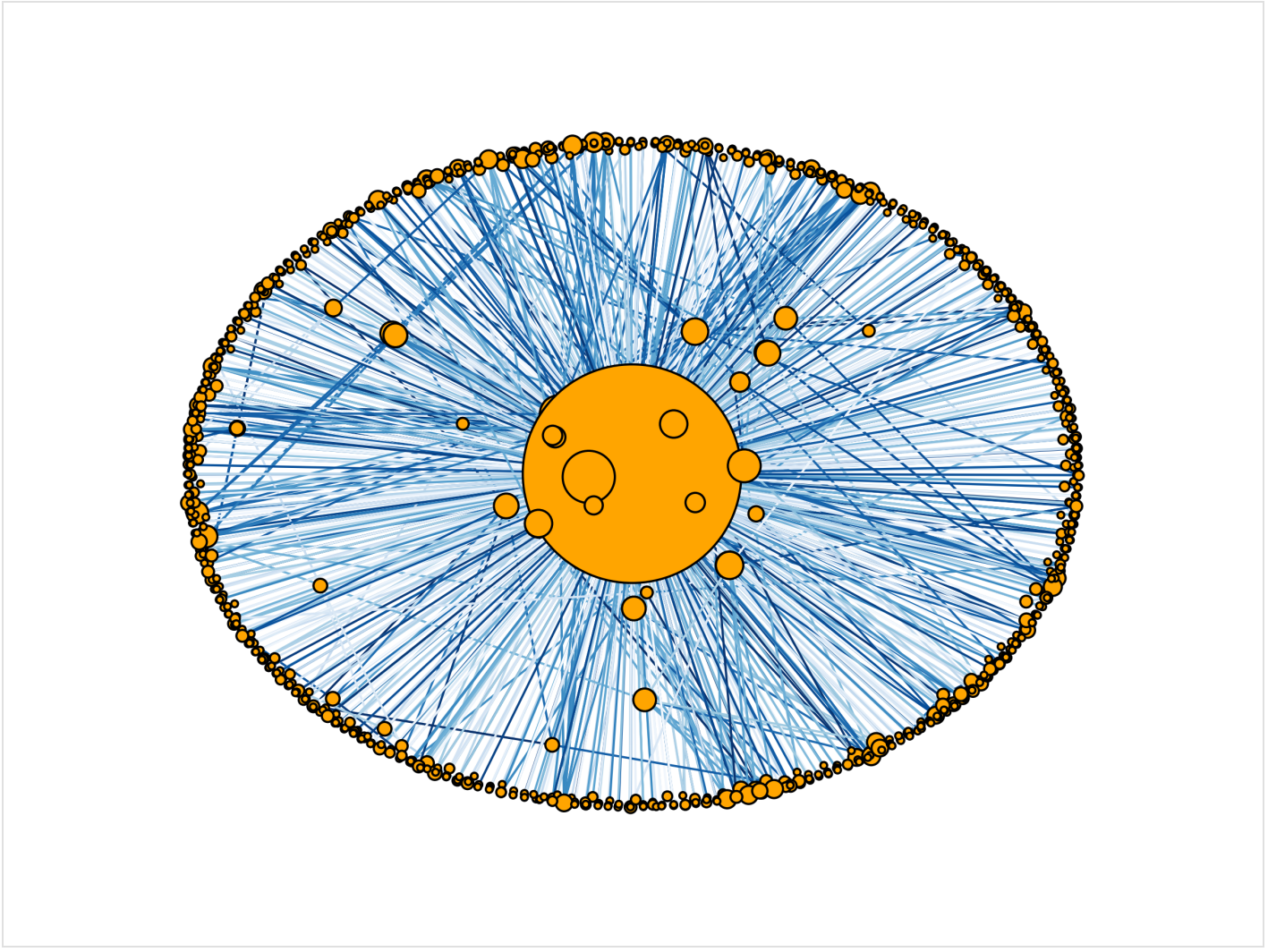


图2：人民日报所发的#云南鲁甸县地震#微博的转发网络

很显然人民日报的这条微博的转发具有明显的星形扩散的特征：与通过社交网络的传播相比，这条微博具有明显的媒体传播特性，即以人民日报作为核心，信息多数是由人民日报直接到达用户，而不需要经过其它用户的中转。

两步流动理论指出信息首先由媒体传递到意见领袖，而后由邮件领袖传播到广大的受众。因此它强调了除了媒体意外，社会网络中的意见领袖对于信息扩散也发挥着重要作用（Lazarsfeld, Berelson, Gaudet, 1944; Katz, E., 1957）。但是，通过这个案例，我们发现并非如此。我们可以认为虽然人民日报官方微博承载在社交网络当中，但是其传播方式依然保持了传播媒体信息的一步到达受众的特点（或许这种特征比线下更强）。

数据分析

对于网络数据的分析，首先是一些网络的统计指标。根据分析的单位分为网络属性、节点属性和传播属性。其中网络属性包括网络的规模，网络群聚系数， 直径和平均距离，匹配性；节点属性包括节点间的距离，中心性等方面；而传播的属性则关注传播的时空和网络特征。

节点属性

就节点的属性而言，我们首先关注节点间的距离。测量了一个节点到网络中所有的其它节点之间的距离，其中最大的距离就是这个节点的离心度（eccentricity）。网络的半径（radius）就是最小的节点离心度；网络的直径（diameter）就是最大的节点离心度。不过，离心度的计算需要将有向网络转化为无向网络。经过计算，该信息转发网络的直径是4，半径是2。

另外，我们可以使用nx.all_pairs_shortest_path_length(G)函数计算任意一对节点之间的路径长度。我们知道源微博的发出者是'2803301701'（即@人民日报）。我们选取另外一个节点'1783628693' 。使用nx.shortest_path来计算两个节点之间的路径。发现节点'1783628693'经过'1904107133'转发源微博，也就是说节点'1783628693'和源微博之间的距离是2。我们还可以计算网络的平均最短距离，发现该有向网络的平均最短路径很小，只有0.001；但如果把网络转化为无向网络，其平均最短路径就大于2了。

```
UG=G.to_undirected()
eccen = nx.eccentricity(UG)#节点离心度
max(eccen.values() #4
min(eccen.values() #2
nx.diameter(G) # 网络直径4
nx.radius(G) #网络半径2
path = nx.all_pairs_shortest_path(G)
nx.shortest_path(G, source = '2803301701', target ='1783628693' )
#['2803301701', '1904107133', '1783628693']
nx.shortest_path_length(G, source = '2803301701', target ='1783628693' ) #2
nx.average_shortest_path_length(G) # 网络平均最短距离0.001
nx.average_shortest_path_length(UG) # 网络平均最短距离2.05
```

另外一个方面，我们关心节点的中心程度。常用的测度包括：节点的度（degree）、接近度(closeness)、中间度(betweenness)。

```
degree = nx.degree(G)
closenesss = nx.closeness_centrality(G)
betweenness = nx.betweenness_centrality(G)
```

对于网络研究而言，一个很重要的方面是关注网络的度分布。现实生活中的大多数网络节点的度具有较强的异质性，即有的节点的度很大，而多数节点的度很小。我们可以画出双对数坐标下的网络度分布以及网络度-排名分布。

```
degree_hist = nx.degree_histogram(G)
x = range(len(degree_hist))
y = [i / int(sum(degree_hist)) for i in degree_hist]
plt.loglog(x, y, color = 'blue', linewidth = 2, marker = 'o')
plt.title('Degree Distribution')
plt.ylabel('Probability')
plt.xlabel('Degree')
plt.show()
```

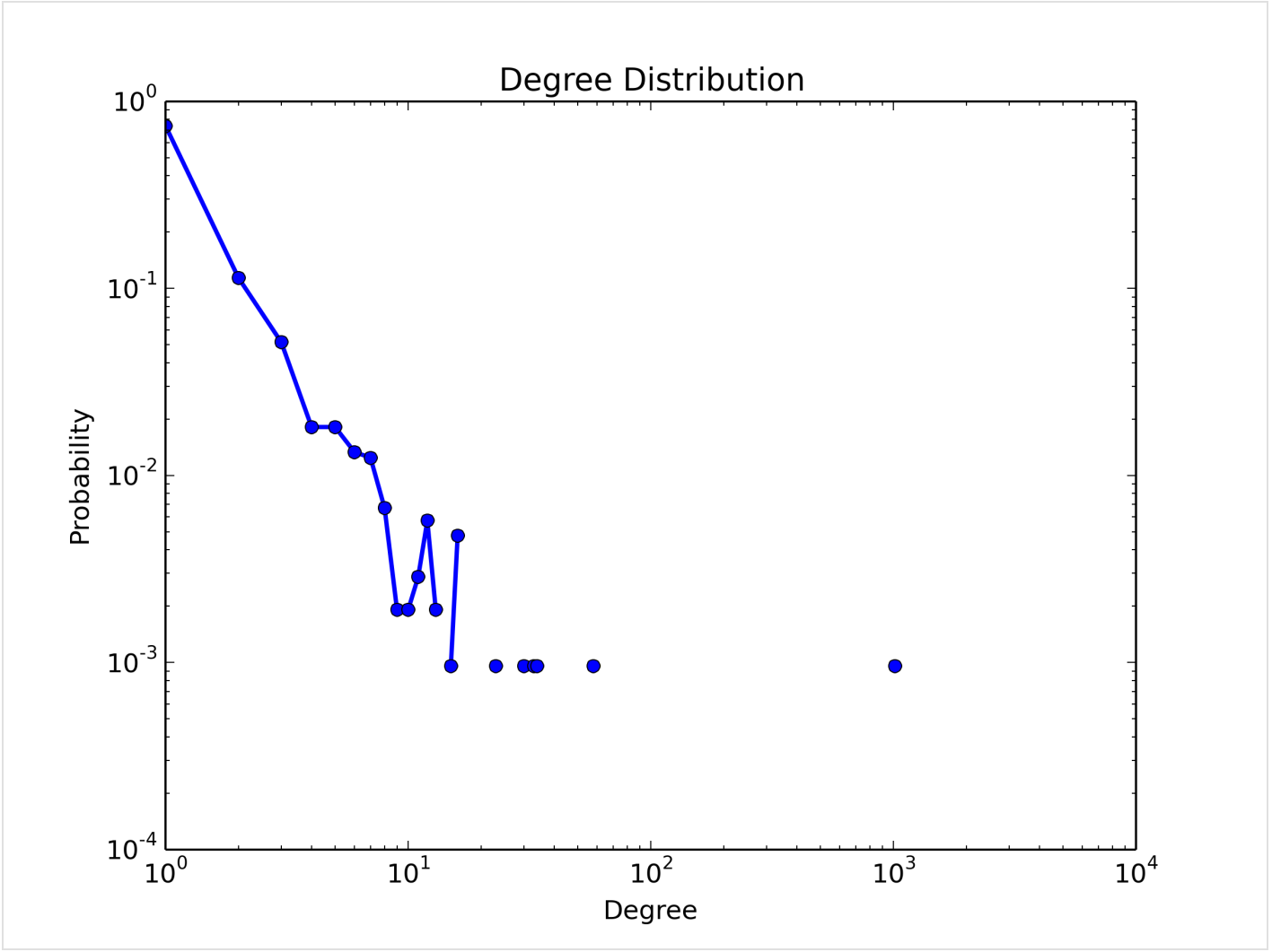


图2：网络度分布

```
# 绘制网络度排名的概率分布图
from collections import defaultdict
import scipy.stats as ss

d = sorted(degree.values(), reverse = True )
d_table = defaultdict(int)
for k in d:
    d_table[k] += 1

d_value = sorted(d_table)
d_freq = [d_table[i] for i in d_value]
d_prob = [i/sum(d_freq) for i in d_freq]
d_rank = ss.rankdata(d_value).astype(int)
plt.loglog(d_rank, d_prob, color = 'blue', linewidth = 2, marker = 'o')
plt.title('Degree Rank-order Distribution')
plt.ylabel('Probability')
plt.xlabel('Rank')
plt.show()
```

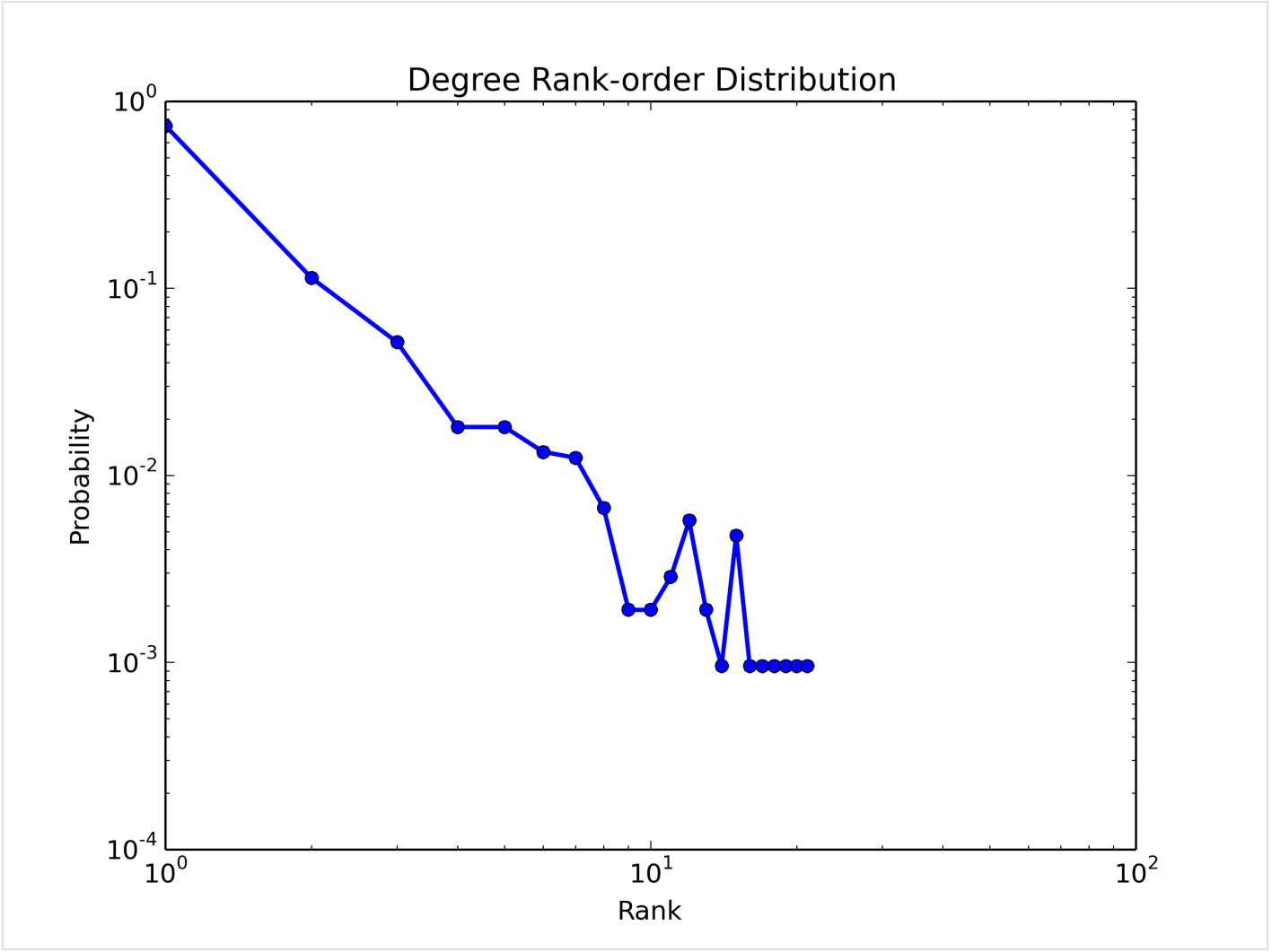


图2：网络度排名概率分布图

网络属性

网络层级的属性使用networkx非常容易计算。根据计算我们发现在这个完整的转发网络当中，共有1047个节点和1508个链接。由此，也可以知道网络的密度（实际存在的链接数量和给定节点的数量可能存在链接数量之间的比值）较小，经过计算只有0.001左右。使用nx.info()函数也可以给出网络节点数量和链接数量。

如果说网络密度关注的是网络中的链接，传递性（transitivity）关注的则是网络中的三角形的数量，传递性也因此被定义为存在的三角形数量与三元组的数量的比值再乘以3（因为一个三角形构成三个未闭合的三元组）。经过计算发现网络传递性的数值是0.001，也非常小，这说明网络中闭合的三角形非常少。

根据节点所在的闭合三角形的数量，还可以计算节点的群聚系数。我们知道，对于没有权重的网络而言，节点的度（D）越高，可能占有的三角形数量（$\frac{D(D-1)}{2}$）就越高。节点的群聚系数就是节点占有的闭合三角形的数量和可能占有的三角形数量之间的比值。使用nx.triangles(G)函数可以计算出每个节点所占有的三角形数量，结合节点的度，就可以计算出节点的群聚系数。当然了，节点群聚系数可以直接使用nx.clustering(G)得到。计算所有网络节点的群聚系数，取其平均值就是网络的群聚系数。经过计算网络的群聚系数为0.227。当然了，网络群聚系数可以直接使用nx.average_clustering(G) 函数得到。另外的一个网络统计指标是匹配性。经过计算，网络节点度的匹配性为负值，即度小的节点多与度大的节点相连。

```
G.number_of_nodes() # 节点数量1047
G.number_of_edges() # 链接数量1508
nx.density(G) # 网络密度0.001
nx.info(G)
#'Name: \nType: DiGraph\nNumber of nodes: 1047\nNumber of edges: 1518\nAverage in degree:   1.4499\nAverage out degree:   1.4499'
nx.transitivity(G) # 传递性0.001
nx.average_clustering(UG) # 网络群聚系数0.227
```

```
nx.degree_assortativity_coefficient(UG) # 匹配性-0.668
```

传播属性

扩散深度

转发者距离原微博发出者的距离可以看做是该条信息转发被中介化的程度。我们已经知道，离心度衡量的是一个节点到其它所有节点距离中的最大值。如果我们测量源微博发出者（@人民日报）的离心度，我们就可以找到这个转发网络的信息扩散深度（diffusion depth）。不难算出，其扩散深度是2。由此可见虽然转发过千人，但是这种扩散主要是广度优先的扩散，其扩散的深度却非常有限。

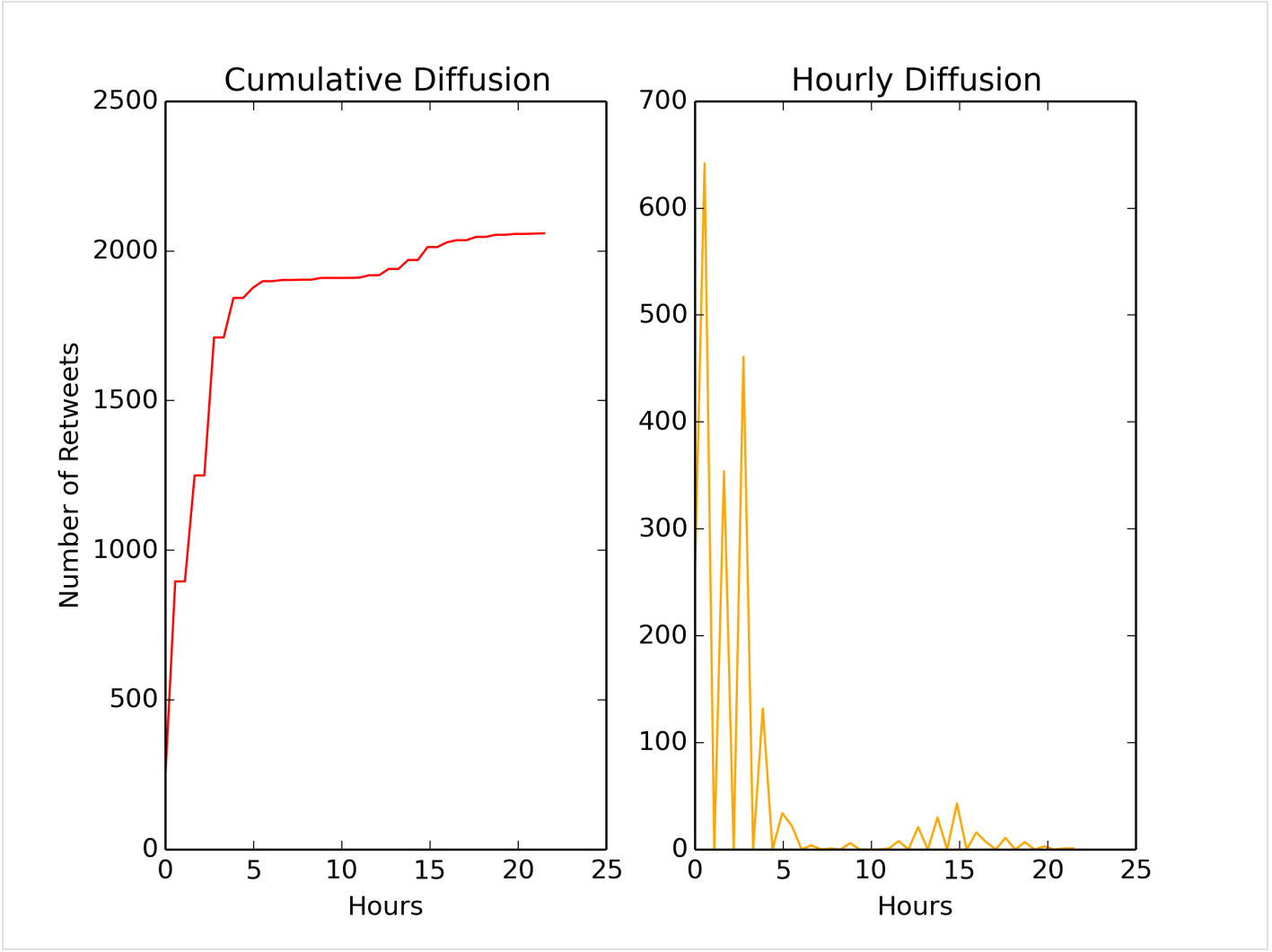
```
eccen = nx.eccentricity(UG) #节点离心度
eccen['2803301701']  #深度为2, '2803301701'即@人民日报
#找到扩散深度最大的节点
path_length = nx.all_pairs_shortest_path_length(G)
oPath = path_length['2803301701']
maxDepth = filter(lambda x: oPath[x] == max(oPath.values()), oPath.keys())
len(maxDepth) # 扩散深度最大的节点数量29
```

扩散速度

```
from datetime import datetime

with open('D:/github/weibo_repost/weibo_repost_network.csv', 'r') as f:
    rt_time = []
    for line in f:
        time= line.strip().split(',')[-1]
        day = time[7:9]
        hms= time[9:17].replace(':', '')
        time = int(day + hms)
        rt_time.append(time)

day = ['2014-08-0'+str(i)[:1]+'-'+str(i)[1:3]) for i in rt_time]
day = [datetime.strptime(d, '%Y-%m-%d-%H') for d in day]
day_weibo = datetime.strptime('2014-08-07-19', '%Y-%m-%d-%H') #源微博发出时间
hours = [(i-day_weibo).total_seconds()/3600 for i in day]
values, base = np.histogram(hours, bins = 40)
cumulative = np.cumsum(values)
plt.subplot(1, 2, 1)
plt.plot(base[:-1], cumulative, c = 'red')
plt.title('Cumulative Diffusion')
plt.ylabel('Number of Retweets')
plt.xlabel('Hours')
plt.subplot(1, 2, 2 )
plt.plot(base[:-1], values, c = 'orange')
plt.title('Hourly Diffusion')
plt.xlabel('Hours')
plt.show()
```



图：微博转发时间分布

源微博发表于2014年8月7日晚上19点，我们统计每一条转发微博的时间与源微博时间的时间差（以小时为单位），结果如上图所示。第一波主要的转发就在源微博发出不久（前5个小时），之后的微博转发速度降低；第二波转发是第二天早上10点左右（第15个小时左右），但是其幅度很低，并且很快降低了。这种模式反应了公众注意力投放的规律，即因为公众注意力有限，所以信息的扩散具有很强的时间限制。以这个微博转发的案例为例，五个小时后，累计增长曲线就开始变得平坦，即使到了第二天中午这种状态也没有得到改变。

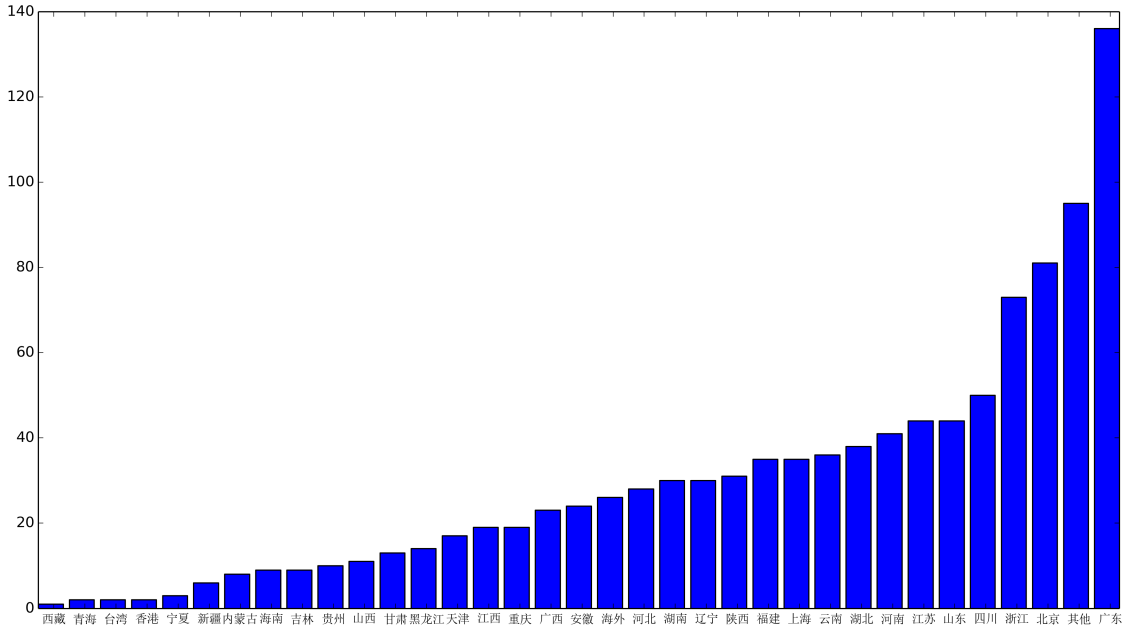
空间分布

此外，我们可以分析转发者的地理分布情况。我们需要提取微博转发者的省份信息。这里还需要新浪微博省市的地理编码。见这里：http://open.weibo.com/wiki/index.php/%E7%9C%81%E4%BB%BD%E5%9F%8E%E5%B8%82%E7%BC%96%E7%A0%81%E8%A1%A8。将这个列表整理为一行代表一个省区的形式，并命名为weibo_province.txt。根据这个地理编码的列表，我们可以将省份编码转化为省份的名称。之后，我们就可以统计各个省区的微博转发数量，并绘制地理分布的直方图。

```
from collections import defaultdict
from matplotlib.font_manager import FontProperties
import numpy as np

#提取微博转发者的省份信息
with open("D:/github/weibo_repost/weibo_repost_all.csv") as f:
    province = dict()
    for line in f:
        line = line.strip().split(',')
        uid = line[7]
        province[uid] = line[9]
```

```
#读取省份代码列表
with open("D:/github/weibo_repost/weibo_province.txt") as f:
    province_dict = dict()
    for line in f:
        pid = line.split('\t')[0]
        provinceName = line.split('\t')[1].split(',')[0]
        province_dict[pid] = provinceName
#获取省份名称
province_info = [province_dict[i] for i in province.values()]
#获取各个省份的微博转发数量
province_table = defaultdict(int)
for k in province_info:
    province_table[k] += 1
#根据微博转发数量对省份名称排序
province_table_sorted = sorted([(value, key) for (key, value) in province_table.items()])
#获取排序后的省份微博转发数量和对应的省份名称
provinceFreq= [i[0] for i in province_table_sorted]
provinceName = [unicode(i[1]) for i in province_table_sorted]
#绘制直方图
position = np.arange(len(provinceName))
ax = plt.axes()
ax.set_xticks(position +0.5)
font = FontProperties(fname = r'c:\windows\fonts\simsun.ttc', size = 10)
ax.set_xticklabels(provinceName, fontproperties = font)
plt.bar(position, provinceFreq)
plt.show()
```



图：微博转发数量的省区分布

由该省区分布可以知道，广东，北京，浙江，四川是转发量最多的四个省份，而内蒙、新疆、西藏、宁夏等西部省份和港澳台等地的转发最少。因为事件的地域相关性，云南省的转发数量也相对较多。

结语

综上所述，本章简单勾勒了使用Python抓取、预处理、分析、可视化社交网络数据的过程。主要以案例为主，其中又以描绘新浪微博单条信息的扩散为主。就数据抓取而言，社会化媒体提供了异常丰富的内容，因此本文所举得例子很容易就可以扩展到更多的案例、更长的时间、更多的网站。可以参阅Russell (2011;2013)在《Mining the Social Web》和《21 Recipes for Mining Twitter》两书中所提供的更多的例子。

就网络分析而言，本文仅仅介绍了一些最基本的分析方法和Python的实现方法，尤其是networkx的使用。值得一提的是，除了Python，还有很多其它的选择，比如R软件；除了networkx之外，还有igraph、graph-tool、Snap.py等其它类库。就涵盖的内容而言，限于篇幅，同样有一些内容没有被囊括进来，诸如网络的生成、网络社区的划分、信息扩散的模拟。

不可否认的是，读者不可能通过本章完全掌握Python的使用、数据的抓取和社交网络研究的分析方法。本书附录中总结了一些常用的资源和工具（软件、类库、书籍等）。读者可根据自己的偏好和研究目的，按图索骥，通过不断地动手练习来达到持续进步的目的。

参考文献

Hetland, Magnus Lie. (2005) Beginning Python: From Novice to Professional. Apress.

Katz, E. (1957). The two-step flow of communication: An up-to-date report on an hypothesis. Public opinion quarterly, 21(1), 61-78.

Paul Felix Lazarsfeld, Bernard Berelson, Hazel Gaudet (1944) The people's choice: How the voter makes up his mind in a presidential campaign. Columbia University Press.

Russell, M. A. (2013). Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More. O'Reilly Media, Inc.

Russell, M. A.(2011). 21 Recipes for Mining Twitter. O'Reilly Media, Inc.