实例：利用现有知识搭建分布式爬虫

分布式爬虫通常运行在多台电脑上，如果没有多台电脑，可以使用在自己的电脑上搭建多台虚拟机，进行分布式爬虫的模拟练习。

一、爬虫架构

分布式爬虫涉及到多个程序的协作，所以在开始编写代码之前需要理清各个程序之间的关系，如下图所示，爬取队列和存储器位于win7主机中，两个下载器位于虚拟机中，其工作步骤如下：

1. 在爬虫开始运行时，下载器从爬取队列中获取需要采集的url；

2. 下载器从url中获取新的url和网页数据；

3. 新的url传回win7主机，加入爬取队列中，加入队列前进行查重；

4. 网页数据传回win7主机进行存储，存储前进行查重；

爬取队列

下载器

下载器

存储器

待爬取url

待爬取url

待爬取url

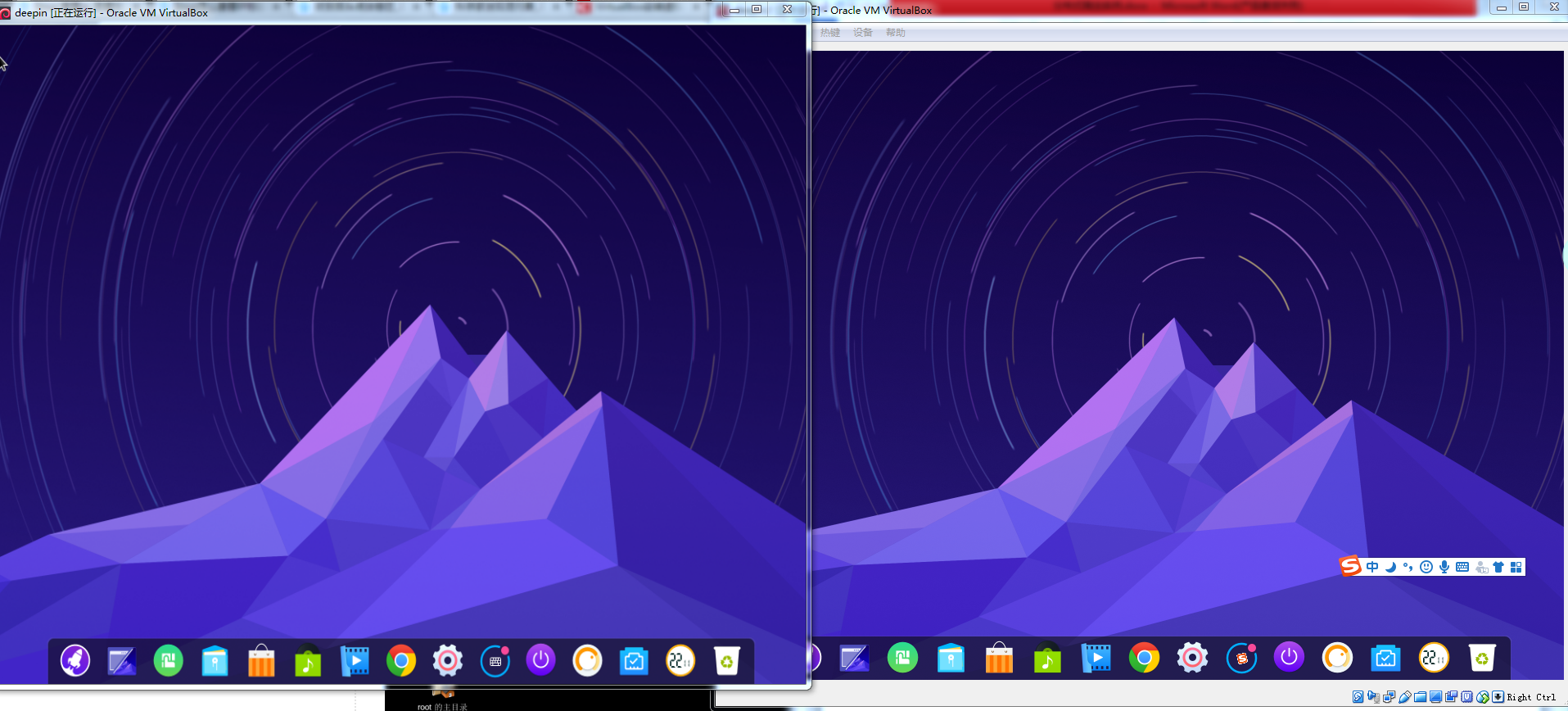
待爬取url

网页数据

网页数据

二、在虚拟机中安装Python环境

本书中虚拟机软件为Oracle VM VirtualBox 5.2.12，虚拟机中系统为deepin 15.3 32bit。



1. 安装Python

Linux系统中一般会自带Python2.X，而Python3.X需要自行安装，在deepin中可以使用apt-get安装Python：

~$ sudo apt-get install python3.4

2. 安装pip

使用上面的命令下载的Python并不带pip,需要另行安装:

~$ wget <https://bootstrap.pypa.io/get-pip.py>

~$ sudo python3 get-pip.py

3. 安装需要的第三方库

~$ sudo python3 –m pip install pymysql

~$ sudo python3 –m pip install tqdm

注意：如非项目需要，不要轻易修改Linux中的默认Python版本。

二、新建两个Mysql远程用户

不需要每台机器都安装Mysql，只需在win7主机上安装，然后虚拟机使用pymysql访问主机的Mysql即可。

但是Mysql默认不允许远程访问，所以在新建远程账户之后要对账户的权限进行设置：

新建账户

mysql> create user worker1 identified by '123456';

mysql> create user worker2 identified by '123456';

授予权限

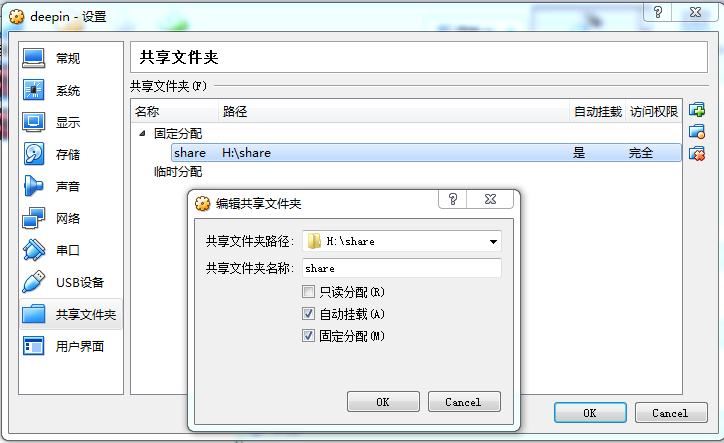
mysql> grant all privileges on \*.\* to worker1@"%" identified by "123456";

mysql> grant all privileges on \*.\* to worker2@"%" identified by "123456";

如果使用命令授权失败的话，可以尝试使用NaviCat或者Mysql Workbench等可视化软件进行授权。

三、虚拟机与主机之间传输文件

1. 设置共享文件夹



2. 安装增强功能

点击虚拟机运行窗口菜单栏中的“设备”选项，点击安装增强功能，虚拟机中会出现VBox\_GAs虚拟光驱，打开虚拟光驱后，在空白处点击：在终端打开，在终端输入命令：

dalalaa@dalalaa-pc:/media/dalalaa/VBox\_GAs\_5.2.12$ sudo ./VBoxLinuxAdditions.run

运行完成之后重启虚拟机

3. 将用户添加到vboxsf组

重启虚拟机之后即可在/media文件夹下看见sf\_share文件夹，但是当前用户对sf\_share文件夹没有权限，需要将用户加入到vboxsf用户组中：

dalalaa@dalalaa-pc:~$ sudo adduser dalalaa vboxsf

运行完成之后重启虚拟机，便可获得hf\_share的权限。

只需在win7主机中的H:/share文件夹中存入文件，便会同步到虚拟机的/media/sf\_share文件夹中。

四、爬虫代码

本爬虫的任务是随机爬取百度贴吧中的帖子，并将帖子名和地址存入数据库。

1. 任务队列

爬虫队列起到调度多个爬虫的作用，需要存储待爬取的url。本爬虫中采取的方式是在Mysql数据库中存储url，待爬取的url状态设为0，已爬取的url状态设为1。

爬虫每次从队列中选取第一个状态为0的url进行爬取，url被提取之后，状态修改为1。

import pymysql

class MysqlQueue():

def \_\_init\_\_(self,conn = None):

self.start\_url = 'tieba.baidu.com/index.html'

self.conn = pymysql.connect(host = "192.168.1.5",port = 3306,user = "worker1",passwd = "123456",db = "spider",charset='utf8') if conn is None else conn

self.cursor = self.conn.cursor()

sql = """

CREATE TABLE IF NOT EXISTS DATA1 (

URL VARCHAR(255) NOT NULL PRIMARY KEY,

USED INT NOT NULL

)CHARSET=utf8;

"""

try:

self.cursor.execute(sql)

except Exception as e:

print(e)

def put(self,url):

# 在插入之前查重

try:

self.cursor.execute("""

INSERT INTO DATA1(

URL,USED

) VALUES (%r,0) ON DUPLICATE KEY UPDATE URL=%r

""" % (url,url))

self.conn.commit()

except Exception as e:

print(e)

def pop(self):

# 提取之后将used字段置为1

self.cursor.execute("""

SELECT URL FROM DATA1 WHERE USED=0 LIMIT 1

""")

url = self.cursor.fetchone()

if url is None:

url = self.start\_url

else:

url = url[0]

self.cursor.execute("""

UPDATE DATA1 SET USED=1 WHERE URL=%r

""" % url)

self.conn.commit()

return url

2. 爬虫主体

from mysql\_queue import MysqlQueue

from mysql\_storage import MysqlStorage

import pymysql

import requests

from lxml import html

from urllib.parse import quote

from tqdm import tqdm

import re

class Spider():

def \_\_init\_\_(self,url=None):

self.que = MysqlQueue() # 队列

self.path = 'D:/data/' # 网页存储地址

self.iterr = 200 # 爬取次数

self.domain = 'tieba.baidu.com' # 爬取的网站主域名

self.storage = MysqlStorage() # 数据存储库

def extract\_urls(self):

'''

从网页中提取所有网址并导入到数据库中

'''

self.url = self.que.pop()

r = requests.get('http://' + self.url)

r.encoding = 'utf-8'

content = r.content # 使用text时会出现解码错误

# 将帖子名存入数据库中，也可以同时将网页存入本地磁盘

if '/p/' in self.url:

# self.download(content)

s = requests.get('https://' + self.url)

tr = html.fromstring(s.text)

titles = tr.xpath('//h1/@title')

if len(titles) > 0:

print(titles)

self.storage.insert(titles[0],quote(self.url))

tree = html.fromstring(content)

url\_list = tree.xpath('//@href')

for url in url\_list:

if 'http://' in url or 'https://' in url or '//' in url:

url = re.sub(r'https://|http://|//','',url)

# 只抓取帖子页面

if re.search(r'^/p/',url): # 帖子

url = self.domain + url

self.que.put(quote(url))

elif re.search(r'^/f\?kw',url): # 帖吧首页

url = self.domain + url

self.que.put(quote(url))

elif 'tieba.baidu.com/' in url:

self.que.put(quote(url))

else:

pass

def run(self):

'''

爬虫主体

'''

for i in tqdm(range(self.iterr)):

try:

self.extract\_urls()

except Exception as e:

print(e)

def download(self,content):

filename = self.url.replace('/','\_') + '.html'

with open(self.path + filename,'wb') as f:

f.write(content)

def request\_url(self):

'''

从库中申请新的url

'''

self.url = self.que.pop()

if \_\_name\_\_ == "\_\_main\_\_":

spider = Spider()

spider.run()

3. 信息存储

import pymysql

class MysqlStorage:

def \_\_init\_\_(self,con=None):

self.conn = pymysql.connect(host = "192.168.1.5",port = 3306,user = "worker1",passwd = "123456",db = "spider",charset='utf8') if con is None else con

self.cursor = self.conn.cursor()

sql = """

CREATE TABLE IF NOT EXISTS DATA2 (

URL VARCHAR(255) NOT NULL PRIMARY KEY,

NAME VARCHAR(255) NOT NULL

)CHARSET=utf8;

"""

try:

self.cursor.execute(sql) # 将url设置为主键，便不能重复

except Exception as e:

print(e)

def insert(self,name,url):

"""

插入数据

"""

sql = """

INSERT INTO DATA2(

NAME,URL

) VALUES (%r,%r) ON DUPLICATE KEY UPDATE NAME= %r;

"""% (name,url,name)

try:

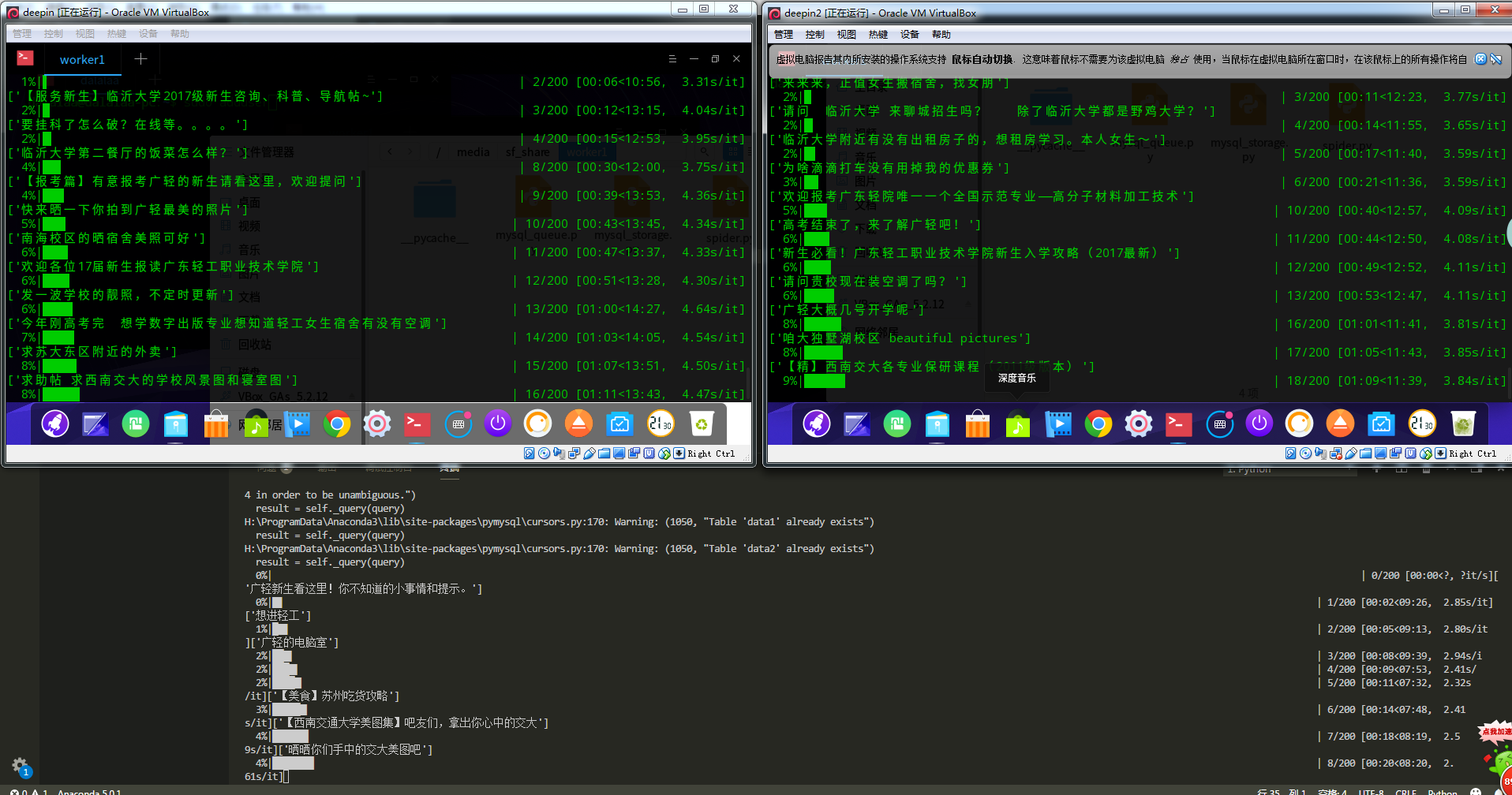
self.cursor.execute(sql)

self.conn.commit()

except Exception as e:

print(e)

按代码编写完成之后，在两台虚拟机中（也可以同时在win7主机中运行爬虫）分别运行爬虫主体，即可实现分布式爬取：



实例：构建合肥美食地图

tableau对国内城市地图的支持不太理想，所以这个实例中的美食地图是使用Matplotlib绘制的，效果可能会比专业的可视化软件更朴素一点。

在专业的地图软件中，区域边界通常是使用坐标点序列的形式表示，如：”116.3288977621845,30.08526472680418;116.42577673164577,29.929492948465658; 116.70319817450891,30.16534338245409;”

1. 绘制区域地图

def draw\_streets():

streets = pd.read\_csv('street\_data.csv')

streets = streets[streets['COORDINATE'].apply(lambda x:True if x is not None else False)]

streets = streets[streets['XZQHSZ\_DM'].apply(lambda x:True if x in ['"340102"','"340103"','"340104"','"340111"'] else False)] # 筛选合肥四区的街道

# 将字符串格式的坐标转化为易于处理的列表形式

def clean(x):

x = re.sub(r'"|;$','',x)

coords = x.split(';')

return [(float(c.split(',')[0]),float(c.split(',')[1])) for c in coords[:-1]] # 最后会出现一个空字符串

streets['COORDINATE'] = streets['COORDINATE'].map(clean)

for s,row in tqdm(streets.iterrows()):

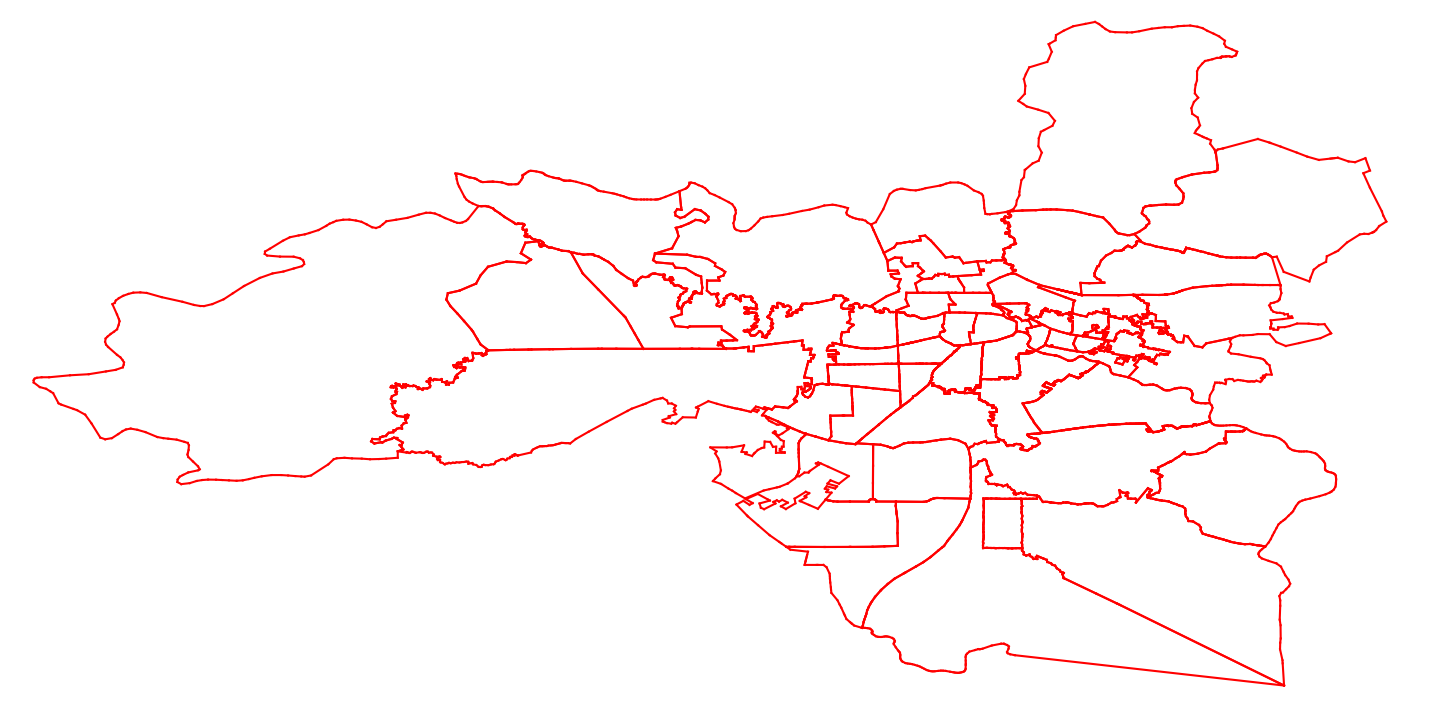
coords = row['COORDINATE']

for i in range(len(coords)-1):

plt.plot([(coords[i][0]),coords[i+1][0]],[coords[i][1],coords[i+1][1]],color = 'r')

plt.show()

return streets



2. 利用百度地图web服务API获取美食地址

百度地图API的调用之前已经介绍过了，这里再温习一下。

下面的这段代码利用了百度地图web服务API的圆形区域检索的功能，请求方式为GET，请求网址为：

http://api.map.baidu.com/place/v2/search?query=银行&location=39.915,116.404&radius=2000&output=json&ak=您的密钥

其中：

query为检索关键词

location为搜索中心点坐标

radius为搜索半径

output为输出格式，可选json或者xml

ak为百度地图api的密钥

检索时，API会自动返回在搜索范围内的前9个最佳匹配项，因此，在大范围地毯式搜索的时候会出现大量的重复地点，在采集时需要进行查重。

①画一个矩形，将合肥地图包围在里面，取四个顶点坐标：

GRID = {

'合肥':[

[116.884841,31.973278],

[116.884841,31.589222],

[117.473626,31.589222],

[117.473626,31.973278]

]

}

②创建一个搜索中心点坐标队列，便于分布式或者多进程协作爬取

class MysqlQueue:

def \_\_init\_\_(self,points,db\_name="food\_map", table\_name="points",conn=None):

self.db\_name = db\_name

self.table\_name = table\_name

self.conn = pymysql.connect("localhost", "worker1", "123456", db\_name) if conn is None else conn

self.cursor = self.conn.cursor()

sql = """

CREATE TABLE IF NOT EXISTS %s (

ID INT NOT NULL PRIMARY KEY AUTO\_INCREMENT,

POINT VARCHAR(255) NOT NULL,

USED INT NOT NULL

)CHARSET=utf8;

""" % table\_name

try:

self.cursor.execute(sql)

except Exception as e:

print(e)

self.points = [(point,0) for point in points]

sql = """

INSERT INTO points(

POINT,USED

) VALUES (%r,%r)

"""

self.cursor.executemany(sql,self.points)

self.conn.commit()

def pop(self):

# 提取之后将used字段置为1

self.cursor.execute("""

SELECT POINT FROM %s WHERE USED=0 LIMIT 1

""" % self.table\_name)

c = self.cursor.fetchone()

if c:

p = c[0] # fetchone会取出一个列表

self.cursor.execute("""

UPDATE %s SET USED=1 WHERE POINT=%r

""" % (self.table\_name, p))

self.conn.commit()

else:

p = None

return p

③创建爬虫，进行美食搜索

class baidu\_spider:

def \_\_init\_\_(self,ak='ttoYeqUv8PCDjdrdXQfLP7M7YcRnO1aQ',keyword='美食',city='合肥',density=150,db\_name = "food\_map",table\_name = 'food\_data',radius = 500,output = 'json',scope = 2,conn=None):

self.ak = ak

self.keyword = keyword

self.city = city

self.density = density

self.radius = radius

self.scope = scope

self.output = output

self.conn = pymysql.connect("localhost", "worker1", "123456", db\_name,charset='utf8') if conn is None else conn

self.cursor = self.conn.cursor()

sql = """

CREATE TABLE IF NOT EXISTS %s (

ID INT NOT NULL AUTO\_INCREMENT,

NAME VARCHAR(255) NOT NULL,

TAG VARCHAR(255),

RATING VARCHAR(255),

DISTRICT VARCHAR(255),

COORDINATE VARCHAR(255) NOT NULL,

COMMENT\_NUM VARCHAR(255),

ADDRESS VARCHAR(255),

TYPE VARCHAR(255),

PRIMARY KEY (ID,NAME,COORDINATE)

)CHARSET=utf8;

""" % table\_name

try:

self.cursor.execute(sql)

except Exception as e:

print(e)

self.table\_name = table\_name

def generate\_points(self):

grid = GRID[self.city]

# 生成搜索中心点坐标

lat = ['%r' % (grid[0][0] + (grid[2][0] - grid[0][0]) \* i / self.density) for i in range(self.density - 1)]

lng = ['%r' % (grid[2][1] + (grid[0][1] - grid[2][1]) \* i / self.density) for i in range(self.density - 1)]

self.points = [n+','+a for a in lat for n in lng]

self.queue = MysqlQueue(points=self.points)

def get\_coordinate(self):

self.generate\_points()

p = self.queue.pop()

i = 0

while p:

if i % 100 == 0:

print(i)

i += 1

url = 'http://api.map.baidu.com/place/v2/search?query=%s&location=%s' \

'&radius=%d&output=%s&scope=%s&ak=%s' % (self.keyword,p,self.radius,self.output,self.scope,self.ak)

r = requests.get(url)

print(r.text)

# 解析返回的json字符串

information = json.loads(r.text)['results']

if information != []:

for infor in information:

name = infor['name']

coordinate = str(infor['location']['lng']) +','+ str(infor['location']['lat'])

district = infor['area']

address = infor['address']

if infor['detail'] == 1:

try:

tag = infor['detail\_info']['tag']

except Exception as e:

tag = ""

try:

is\_type = infor['detail\_info']['type']

except Exception as e:

is\_type = ""

try:

rating = infor['detail\_info']['overall\_rating']

except Exception as e:

rating = ""

try:

comment\_num = infor['detail\_info']['comment\_num']

except Exception as e:

comment\_num = ""

else:

tag,is\_type,rating,comment\_num = "","","",""

sql = """

INSERT INTO %s(

NAME,TAG,RATING,DISTRICT,

COORDINATE,COMMENT\_NUM,ADDRESS,TYPE

) VALUES (%r,%r,%r,%r,%r,%r,%r,%r) ON DUPLICATE KEY UPDATE NAME=%r;

""" % (self.table\_name,name,tag,rating,district,coordinate,comment\_num,address,is\_type,name)

self.cursor.execute(sql)

self.conn.commit()

p = self.queue.pop() # 重新获取下一个搜索中心点p

3. 数据分析

得到的数据如下表所示：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | TYPE | NAME | COORDINATE | DISTRICT | RATING | COMMENT\_NUM | ADDRESS | TAG |
| 1 | cater | \*饭店 | "116.839299,31.905239" | 蜀山区 | 4.5 | 5 | 合肥市蜀山区\*\*\*\* | 美食；中餐厅 |

可以根据坐标将这些饭店绘制在上面的合肥四区地图上。

在绘制地图的代码中plt.show()上方添加如下代码：

con = pymysql.connect(host="localhost",user="worker1",password="123456",db="food\_data")

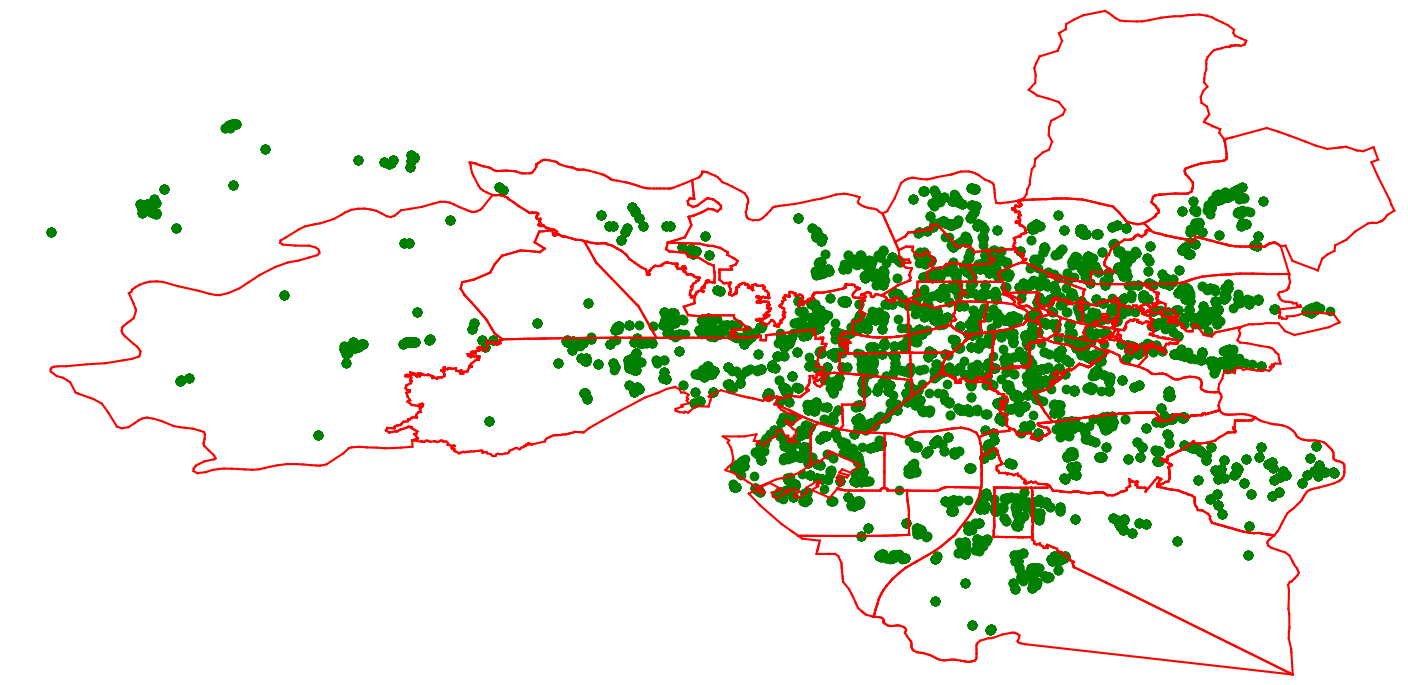
res = pd.read\_sql("select \* from food\_data",con)

res = res[res['XZQHSZ\_DM'].apply(lambda x: True if x in ["蜀山区","包河区","庐阳区","瑶海区"] else False)]

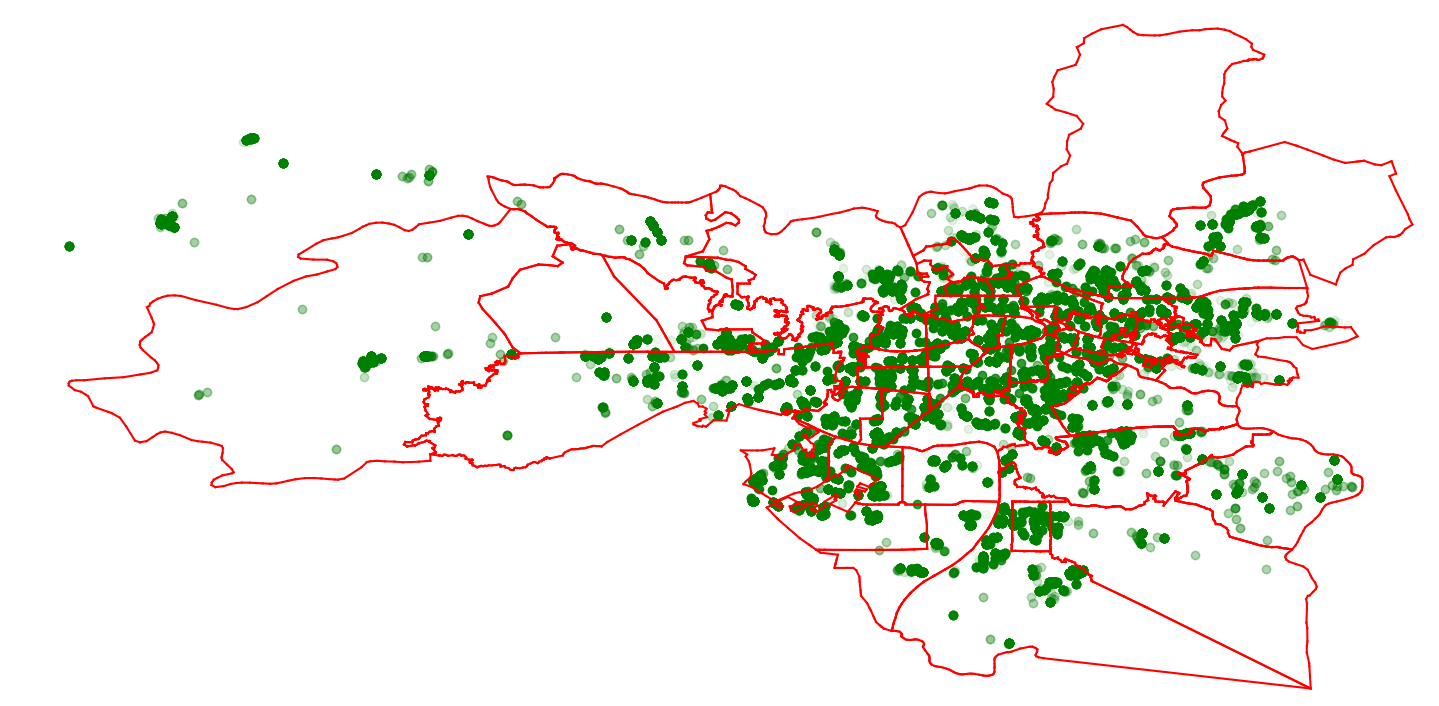
for c in res['COORDINATE'].values:

plt.scatter(float(c.split(',')[0]),float(c.split(',')[1]),color= 'g')

即可得到如下图像：



这样便能从地图中看到合肥四区中的美食分布密度。为了方便地查看合肥各地区的美食质量，还需要根据美食评分采用不同的颜色深浅来绘制美食点，将绘制美食点的代码修改如下：



展示效果仍然不理想，可以考虑采用热力图的方式进行展示，根据街道中的美食平均分对街道填充不同深度的颜色，绘制热力图还需要两个辅助功能：

1. 判断点是否在多边形内部

判断点是否在多边形内部可以使用射线法，即在图中任选一点，作一条射线，如果该射线与多边形的交点数量为奇数，则可以判定点在多边形内部，如果是偶数则可以判定多边形在外部。

def pointInPolygon(point,polygon):

j = len(polygon) - 1

result = False

for i in range(len(polygon)):

p1 = polygon[i]

p2 = polygon[j]

#如果point正好是polygon上的顶点

if (p1[0] == point[0] and p1[1] == point[1]) or (p2[0] == point[0] and p2[1] == point[1]):

return True

j = i

if point[1] > min(p1[1],p2[1]):

if point[1] <= max(p1[1],p2[1]):#如果point的纵坐标在线段p1p2的范围内

if point[0] < max(p1[0],p2[0]):

if p1[1] != p2[1]:#如果线段p1p2不是水平线段，则求过point的水平直线与线段p1p2的交点

x = (point[1]-p1[1])\*(p2[0]-p1[0])/(p2[1]-p1[1]) + p1[0]

if p1[0] == p2[0] or point[0] < x:

result = not result

if point[0] == x:

return True

return result

2. 多边形填充

多边形填充功能采用了扫描线填充算法，即先从任一点出发画直线，求直线与多边形之间的交点，然后再通过在交点之间画粗线的方式填充多边形。

def cross\_point(polygon,num):

'''polygon是多边形点列表，num是水平线的y坐标

return 交点坐标

'''

ps = []

for i in range(len(polygon)-1):

p1 = polygon[i]

p2 = polygon[i+1]

if max(p1[1],p2[1]) < num or min(p1[1],p2[1]) > num:

continue

#如果p1p2不是水平线

if p1[1] != p2[1]:

x = (num-p1[1])\*(p2[0]-p1[0])/(p2[1]-p1[1])+p1[0]

#如果p1p2是水平线，水平线不求交点

if p1[1] == p2[1]:

x = None

#左闭右开，但是要排除竖直线的情况，因为竖直线时，x==p1[0] and x==p2[0]

#if x == p2[0] and x != p1[0]:

if x== p2[0] and num == p2[1]:

x = None

#判断是不是上下顶点

if i == 0:

p0 = polygon[len(polygon)-2]

else:

p0 = polygon[i-1]

#如果是上下顶点

if x == p1[0]:

if (p0[1]>p1[1] and p2[1]>p1[1]) or (p0[1]<p1[1] and p2[1]<p1[1]):

x = None

if x != None:

if x==2.0:

print(p1[0],p1[1],p2[0],p2[1])

ps.append((x,num))

return ps

def fill(df,col = 'AVG\_RATING'):

'''填充'''

nums = [31.5000+i/3000 for i in range(3000)]

#df['COORDINATE'] = df['COORDINATE'].map(seg\_to\_point)

#for poly in tqdm(list(df['COORDINATE'].map(seg\_to\_point).values)):

#红黄绿篮紫

for i,row in tqdm(df.iterrows()):

poly = row['COORDINATE']

alpha = 0.5

if row['AVG\_RATING']>= 4.5:

color = 'red'

elif row['AVG\_RATING']> 4.3 and row['AVG\_RATING']< 4.5:

# color = 'orangered'

color = 'tomato'

elif row['AVG\_RATING']> 4.2 and row['AVG\_RATING']<=4.3:

# color = 'tomato'

# color = 'yellow'

color = 'orange'

elif row['AVG\_RATING']> 4.1 and row['AVG\_RATING']<=4.2:

# color = 'salmon'

# color = 'green'

color = 'yellow'

elif row['AVG\_RATING']> 3.9 and row['AVG\_RATING']<=4.1:

# color = 'lightsalmon'

color = 'greenyellow'

else:

# color = 'mistyrose'

color = 'lightblue'

for num in nums:

cps = cross\_point(poly,num)

for i in range(int(len(cps)/2)):

line = (cps[2\*i],cps[2\*i + 1])

plt.plot([line[0][0],line[1][0]],[line[0][1],line[1][1]],color=color,linewidth=3,alpha=alpha)添加完这两个功能之后就可以开始绘制热力图了，绘制热力图首先要对原始数据进行处理，计算平局评分AVG\_RATING。

import pandas as pd

import matplotlib.pyplot as plt

from tqdm import tqdm

from point\_in\_polygon import pointInPolygon

from fill import fill

import re

df = pd.read\_csv('C:\\Users\\Dl\\Documents\\GitHub\\learning\_notes\\food\_map\\new\_items.csv')

df['RATING'] = df['RATING'].fillna(df['RATING'].mean())

df['COORDINATE'] = df['COORDINATE'].apply(lambda x:eval('('+str(x) + ')'))

streets = pd.read\_csv('C:\\Users\\Dl\\Documents\\GitHub\\learning\_notes\\food\_map\\street\_data.csv')

streets = streets[streets['COORDINATE'].apply(lambda x:True if x is not None else False)]

streets = streets[streets['XZQHSZ\_DM'].apply(lambda x:True if x in ['"340102"','"340103"','"340104"','"340111"'] else False)]

# 将字符串格式的坐标转化为易于处理的列表形式

def clean(x):

x = re.sub(r'"|;$','',x)

coords = x.split(';')

return [(float(c.split(',')[0]),float(c.split(',')[1])) for c in coords[:-1]] # 最后会出现一个空字符串

streets['COORDINATE'] = streets['COORDINATE'].map(clean)

# 匹配美食所属街道

belong\_street = []

for i,row1 in tqdm(df.iterrows()):

b\_street = ""

for j,row2 in streets.iterrows():

if pointInPolygon(row1['COORDINATE'],row2['COORDINATE']):

b\_street = row2['QH\_CODE']

break

belong\_street.append(b\_street)

df['QH\_CODE'] = belong\_street

# 计算每个街道的平均评分

avg\_ratings = []

for h,row in tqdm(streets.iterrows()):

code = row['QH\_CODE']

avg\_ratings.append(round(df[df['QH\_CODE']== code]['RATING'].mean(),1))

streets['AVG\_RATING'] = avg\_ratings

print(streets['AVG\_RATING'].value\_counts())

# 填充

fill(streets)

# 绘制地图边界

for s,row in tqdm(streets.iterrows()):

coords = row['COORDINATE']

for i in range(len(coords)-1):

plt.plot([(coords[i][0]),coords[i+1][0]],[coords[i][1],coords[i+1][1]],color = 'black')

plt.show()

运行代码，最终得到的合肥美食地图如下图所示：

