相关教材：林子雨、郑海山、赖永炫编著《Spark编程基础（Python版）》（[访问教材官网](http://dblab.xmu.edu.cn/post/spark-python/" \t "_blank)）  
相关案例：[基于Python语言的Spark数据处理分析案例集锦（PySpark）](http://dblab.xmu.edu.cn/blog/2738-2/" \t "_blank)

本案例以2020年美国新冠肺炎疫情数据作为数据集，以Python为编程语言，使用Spark对数据进行分析，并对分析结果进行可视化。

# 一、实验环境

（1）Linux： Ubuntu 16.04  
（2）Hadoop3.1.3

（3）Python: 3.6  
（4）Spark: 2.4.0

（5）Jupyter Notebook

# 二、数据部分

#### 1.数据爬取

1.1目标网页百度疫情报告



我的最初想法认为数据是通过异步加载，通过分析结构发现数据就在网页中，只是它位于js代码中，是通过js定位然后传入数据。

我直接通过xpath定位数据id到数据所在位置

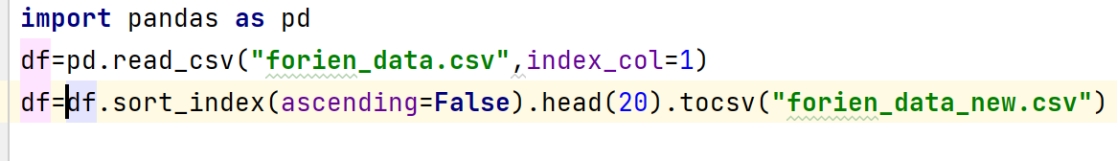


但是该出数据量很大，经过json化和长时间的结构分析，得出国内的数据在

国外数据



但是要求是国外前二十的数据，并没有在结构中发现，于是拿出所有的国外疫情数据，在后续编写代码进行提取。



以上拿出所有的国内外数据,下一步通过seaborn和matpoltlib进行数据的可视化展示

1.2代码部分

from lxml import etree

import json

import requests

import pandas as pd

#爬取的网址（百度疫情）

url="https://voice.baidu.com/act/newpneumonia/newpneumonia/?from=osari\_pc\_3"

#伪装请求头

headers ={'User-Agent':'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36'}

#获取网页地址

response=requests.get(url,timeout=30,headers=headers)

print(response.text)

html=etree.HTML(response.text)

#在网页中寻找我们想要的数据（可以找到对应标签右键复制xpath）

result=html.xpath('//\*[@id="captain-config"]/text()')

#print(result)

#result[0]不是真正的字典类型，而是json字符串

#需要通过json.loads转化json字符串为Python的字典类型

result=json.loads(result[0])

#print(result)

#一层一层找自己所用到的数据

result\_out=result["component"][0]

print(result\_out)

#获取国内疫情数据（找到了

result\_in=result["component"][0]["caseList"]

# print(result\_in)

ws=[]

#用append生成表第一行

time=result["component"][0]['mapLastUpdatedTime']

#循环遍历数据，按照位置顺序，将数据加入到excel中

for each in result\_in:

temp\_list=[each['area'],each['confirmed'],each['died'],each['crued'],each['confirmedRelative']]

ws.append(temp\_list)

ws\_df=pd.DataFrame(ws)

ws\_df.to\_csv("china\_data.csv",mode="a", index=False, encoding='utf\_8\_sig',header=['国家','累计确诊','死亡','治愈','新增'])

#获取国外疫情数据

result\_for=result["component"][0]['caseOutsideList']

fws=[]

#循环遍历数据，按照位置顺序，将数据加入到excel中

for each in result\_for:

temp\_list=[each['area'],each['confirmed'],each['died'],each['crued'],each['confirmedRelative']]

fws.append(temp\_list)

fws\_df = pd.DataFrame(fws)

fws\_df.to\_csv("forien\_data.csv", mode="a", index=False, encoding='utf\_8\_sig',header=['国家','累计确诊','死亡','治愈','新增'])

自此，我们获得了当日内国内所有省份和所有国外的疫情情况

#### 2. pyspark数据处理

原始数据集是以.csv文件组织的，为了方便spark读取生成RDD或者DataFrame，读取数据为dataframe，通过累计确诊排序取出前十个国家。

具体代码如下：

1. from pyspark.sql import SparkSession  
   import findspark
2. findspark.init()  
   spark=SparkSession.builder.appName("data\_processing").getOrCreate()
3. df=spark.read.csv(**'forien\_data.csv'**,inferSchema=**True**,header=**True**).sort([**"累计确诊"**]).collect()[:10]

# 数据可视化

## 1.1 可视化工具选择与代码

选择使用python第三方库pyecharts作为可视化工具。  
在使用前，需要安装pyecharts，安装代码如下：

1. pip install pyecharts

具体可视化实现代码组织与定义一个draw\_Radar函数，对得到的数据分别采集进入数据内容，生成radar\_chart.html

具体代码如下：

1. def draw\_Radar(data\_radar):  
    scope=[]
2. for col in range(1, data\_radar.shape[1]):
3. dic = {}
4. dic['name'] = data\_radar.columns[col]  
    dic['max'], dic['min'] = max(data\_radar.iloc[
5. :, col]), 0  
    scope.append(dic)  
    c = (
6. # Radar()
7. Radar(init\_opts=opts.InitOpts(bg\_color={"type": "pattern", "image": JsCode("img"), "repeat": "no-repeat"}))  
    .add\_js\_funcs(""" var img = new Image();  
    img.src = '背景.jpg'; """)  
    .add\_schema(
8. schema=scope,  
    shape="circle",  
    center=["50%", "50%"], # 宽高：900px\*500px  
    radius="60%",  
    angleaxis\_opts=opts.AngleAxisOpts(  
    min\_=0,  
    max\_=360,  
    is\_clockwise=False,  
    interval=10,  
    axistick\_opts=opts.AxisTickOpts(is\_show=True),  
    axislabel\_opts=opts.LabelOpts(is\_show=True),  
    axisline\_opts=opts.AxisLineOpts(is\_show=True),  
    splitline\_opts=opts.SplitLineOpts(is\_show=True)  
    ),  
    radiusaxis\_opts=opts.RadiusAxisOpts(
9. min\_=0,  
    max\_=30,  
    interval=5,  
    splitarea\_opts=opts.SplitAreaOpts(  
    is\_show=True, areastyle\_opts=opts.AreaStyleOpts(opacity=1)  
    ),  
    ),  
    polar\_opts=opts.PolarOpts(),  
    splitline\_opt=opts.SplitLineOpts(is\_show=False)  
    )  
    .add(  
    series\_name="美国",  
    data=data\_radar[0],
10. areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#fc5a50'  
     )  
     .add(  
     series\_name="印度",  
     data=data\_radar[1],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#35ad6b'  
     )  
     .add(  
     series\_name="巴西",  
     data=data\_radar[2],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#3d7afd'  
     )  
     .add(  
     series\_name="法国",  
     data=data\_radar[3],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#aa23ff'  
     )  
     .add(  
     series\_name="德国",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#b865c6'
11. )  
     .add(  
     series\_name="英国",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#fcb001'  
     )  
     .add(  
     series\_name="俄罗斯",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#4cc1f5'  
     )  
     .add(  
     series\_name="韩国",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2)
12. ,  
     color='#fcb001'  
     )  
     .add(  
     series\_name="意大利",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#fd854a'  
     )  
     .add(  
     series\_name="土耳其",  
     data=data\_radar[4],  
     areastyle\_opts=opts.AreaStyleOpts(opacity=0.2),  
     linestyle\_opts=opts.LineStyleOpts(width=2),  
     color='#ED4D83'  
     )  
      
     .set\_series\_opts(label\_opts=opts.LabelOpts(is\_show=False))  
     .set\_global\_opts(title\_opts=opts.TitleOpts(title='国外疫情情况', subtitle='国家', pos\_left='left'))  
     .render("radar\_chart.html")

## 1.2 运行主程序，得到radar\_chart.html网页文件

1. **if** \_\_name\_\_ == **'\_\_main\_\_'**:  
    df=spark.read.csv(**'forien\_data.csv'**,inferSchema=**True**,header=**True**).sort([**"累计确诊"**]).collect()[:10]  
    draw\_Radar(df)

## 2.1 结果图展示

