#!/usr/bin/env python

*#-\*- encoding: utf-8 -\*-*

**import** **requests**

**import** **time**

**import** **re**

**from** **lxml** **import** etree

**import** **pandas** **as** **pd**

*#import webbroser*

**import** **numpy** **as** **np**

**from** **matplotlib** **import** font\_manager **as** fm, rcParams

**import** **matplotlib.pyplot** **as** **plt**

*#import os*

*#import random*

*#import json*

**from** **urllib.request** **import** urlopen, quote

**import** **csv**

*#import re, pandas as pd*

**from** **sklearn.preprocessing** **import** StandardScaler

**from** **io** **import** BytesIO

**import** **base64**

*#爬取数据过程*

**def** is\_all\_chinese(strs):

**for** \_char **in** strs:

**if** **not** '**\u4e00**' <= \_char <= '**\u9fa5**':

**return** **False**

**return** **True**

**def** getList():

start\_url = 'https://book.douban.com/tag/青春?&type=T&start=**{}**'

i = 0

infoList = []

print("start")

**while** i < 6:

print(i)

url = start\_url.format(i\*20)

kv = {'User-Agent': 'Mozilla/5.0'}

r = requests.get(url,headers=kv)

selector = etree.HTML(r.text)

i += 1

informations = selector.xpath('.//ul[@class="subject-list"]/li')

**for** inf **in** informations:

info\_bookname = inf.xpath('normalize-space(.//div[@class="info"]/h2/a/@title)')

info\_str = inf.xpath('normalize-space(.//div[@class="pub"]/text())')*#[0].replaceAll("\r|\n", "")*

info\_author = ''

strlist = info\_str.split('/')

k = 0

index = 0

**while** k < len(strlist):

**if**("出版" **in** strlist[k]):

index = k

**break**

k += 1

**if**(k == len(strlist)):

info\_publisher = '\*\*\*'

s=0

index1 = 0

**while** s < len(strlist):

**if**("." **in** strlist[s]):

info\_price = strlist[s]

*#index1 = s*

index = s-1

**break**

s += 1

**if**(s == len(strlist)):

info\_price = strlist[len(strlist)-1]

index = len(strlist)-2

**else**:

info\_publisher = strlist[index]*#.replaceAll("\r|\n", "")*

info\_price = strlist[index+2]*#.replace('.','')#.replaceAll("\r|\n", "")*

j = 0

**while** j < index:

info\_author += strlist[j]

j += 1

**if**("元" **not** **in** info\_price):

info\_price = info\_price+"元"

info\_score = inf.xpath('normalize-space(.//span[@class="rating\_nums"]/text())')*#[0].replaceAll("\r|\n", "")*

info\_commentsnum = inf.xpath('normalize-space(.//span[@class="pl"]/text())').replace('(','')

info\_commentsnum = info\_commentsnum.replace(')','')

info\_commentsnum = info\_commentsnum.replace('评价','')

info\_paper\_edi\_price = inf.xpath('normalize-space(.//span[@class="buy-info"]/a/text())')

paper = '纸质版'

**if**(paper **not** **in** info\_paper\_edi\_price):

info\_paper\_edi\_price = '\*\*\*'

**else**:

info\_paper\_edi\_price = info\_paper\_edi\_price.replace('纸质版 ','')

**if**(is\_all\_chinese(info\_bookname)):

infoList.append([info\_bookname,info\_author,info\_publisher,info\_price,info\_paper\_edi\_price,info\_score,info\_commentsnum])

time.sleep(10)

**return** infoList

*#数据持久化（.csv）*

**def** listToCsv(list,file):

df = pd.DataFrame(list,columns=['书名','作者','出版社','售价','纸质版售价','豆瓣评分','评论数量'])

df.to\_csv(file, line\_terminator="**\n**", index=**False**)

file.close()

print(df)

**return** list

In [2]:

*#可视化*

*#豆瓣书籍中各个评分所占比例*

**def** ranks(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

df\_s=df.sort\_values(by=['豆瓣评分'],ascending=**False**)

df\_info=df\_s.iloc[0:20]*#评分前20的书籍信息*

df\_info.to\_csv('评分前20的书籍信息.csv')

mask = df\_info[(**True**^df['豆瓣评分'].isin([0]))]

mask['Mask'] = 1*#添加计数器*

df\_mask = mask[['豆瓣评分','Mask']]

df\_mask1 = df\_mask.groupby(['豆瓣评分']).sum()*#按照级别进行分组，得到不同评分数量*

plt.rcParams['font.sans-serif']=['SimHei'] *#显示中文标签*

plt.rcParams['font.size'] = 12 *# 字体大小*

plt.rcParams['axes.unicode\_minus'] = **False** *# 正常显示负号*

df\_mask1.plot.pie(subplots = **True**, figsize = (6,6))*#画饼图*

*# figure 保存为二进制文件*

img = BytesIO()

plt.savefig(img,format='png',bbox\_inches='tight')

img.seek(0)

**return** base64.b64encode(img.getvalue())

*#评分最高的十本小说的豆瓣评分*

**def** comment\_high(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

df\_s=df.sort\_values(by=['豆瓣评分'],ascending=**False**)

df\_ss=df\_s.iloc[0:20]*#评分前20的书籍信息*

df\_ss.to\_csv('评分前20的书籍信息.csv')

df\_maskF=df\_ss.iloc[0:10]

df\_Firstm=df\_maskF[['书名','豆瓣评分']]

plt.rcParams['font.sans-serif']=['SimHei'] *#显示中文标签*

plt.rcParams['font.size'] = 12 *# 字体大小*

plt.rcParams['axes.unicode\_minus'] = **False** *# 正常显示负号*

df\_Firstm.plot(kind = 'hist',title='评分最高的书籍的豆瓣评分（前10） 分布频率柱形图') *#分布频率柱形图*

ax = plt.gca()

ax.axes.get\_xaxis().set\_visible(**False**)

writer = [i[0] **for** i **in** df\_Firstm[['书名']].values]

ranks = [i[0] **for** i **in** df\_Firstm[['豆瓣评分']].values]

x = [0,1,2,3,4,5,6,7,8,9]

rank\_tmp=0

y\_tmp=-0.5

**for** i **in** x:

**if** rank\_tmp!=ranks[i]:

y\_tmp=-0.5

plt.text(ranks[i],y\_tmp, '**%s**' %**writer**[i], ha='center',rotation=-20, fontsize=12)

rank\_tmp=ranks[i]

**else**:

y\_tmp=y\_tmp-0.5

plt.text(ranks[i],y\_tmp, '**%s**' %**writer**[i], ha='center',rotation=-20, fontsize=12)

*# figure 保存为二进制文件*

img = BytesIO()

plt.savefig(img,format='png',bbox\_inches='tight')

img.seek(0)

**return** base64.b64encode(img.getvalue())

*#收录的作品最多的前五名作者*

**def** auther(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

author = df[(**True**^df['作者'].isin([0]))]

author['aut\_num'] = 1*#添加计数器*

df\_author = author[ ['作者','aut\_num'] ]

df\_author1 = df\_author.groupby(['作者'])

df\_author2 = df\_author1.sum()*#按照作者进行分组，得到不同作者的图书数量*

df\_auth=df\_author2.sort\_values(by=['aut\_num'],ascending=**False**)

df\_auinfo=df\_auth.iloc[0:5]

plt.rcParams['font.sans-serif']=['SimHei'] *#显示中文标签*

plt.rcParams['font.size'] = 12 *# 字体大小*

plt.rcParams['axes.unicode\_minus'] = **False** *# 正常显示负号*

df\_auinfo.plot(kind = 'barh',color='green',width=0.8,alpha=0.5,title='收录最多的书籍作者（前5）柱形图')*#青春小说书籍中 收录最多的前五位作者的柱形图*

*# figure 保存为二进制文件*

img = BytesIO()

plt.savefig(img,format='png',bbox\_inches='tight')

img.seek(0)

**return** base64.b64encode(img.getvalue())

*#青春小说书籍中 小说的评论数量*

**def** comment\_num(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

df['cmt\_num']=df['评论数量'].str.replace(r'[^0-9]','')

df['cmt\_num']=pd.to\_numeric(df['cmt\_num'])

df['cmt\_num']=df['cmt\_num'].round(0).astype(int)

df\_cmtsort=df.sort\_values(by=['cmt\_num'],ascending=**False**)

df\_cmtmost=df\_cmtsort[['书名','cmt\_num']]

plt.rcParams['font.sans-serif']=['SimHei'] *#显示中文标签*

plt.rcParams['font.size'] = 12 *# 字体大小*

plt.rcParams['axes.unicode\_minus'] = **False** *# 正常显示负号*

df\_cmtmost.plot(kind = 'barh',align='center',alpha=0.5,title='书籍的评论总数柱形图')*#青春小说书籍中 评论数量 柱形图*

ax = plt.gca()

ax.axes.get\_yaxis().set\_visible(**False**)

*#ax.spines['left'].set\_visible(False)*

*# figure 保存为二进制文件*

img = BytesIO()

plt.savefig(img,format='png',bbox\_inches='tight')

img.seek(0)

**return** base64.b64encode(img.getvalue())

**def** comment\_num2(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

df['cmt\_num']=df['评论数量'].str.replace(r'[^0-9]','')

df['cmt\_num']=pd.to\_numeric(df['cmt\_num']).round(0).astype(int)

df\_cmtsort=df.sort\_values(by=['cmt\_num'],ascending=**False**)

df\_cmtmost=df\_cmtsort.iloc[0:20]*#评论数量最多前20的书籍信息*

df\_cmtmost.to\_csv('评论数量最多前20的书籍信息.csv')

df\_comment = pd.read\_csv("评论数量最多前20的书籍信息.csv")

df\_comment1=df\_comment[['书名','cmt\_num']]

plt.rcParams['font.sans-serif']=['SimHei'] *#显示中文标签*

plt.rcParams['font.size'] = 12 *# 字体大小*

plt.rcParams['axes.unicode\_minus'] = **False** *# 正常显示负号*

df\_comment1.plot(marker='\*', ms=10,linewidth=2.0,linestyle='--',title='评论数量最多的书籍评论总数折线图（前20）')

writer = [i[0] **for** i **in** df\_comment1[['书名']].values]

fre = [i[0] **for** i **in** df\_comment1[['cmt\_num']].values]

x = [0,1,2,3]

**for** i **in** x:

plt.text(x[i],fre[i], '**%s**' %**writer**[i], ha='center', fontsize=12)

*# figure 保存为二进制文件*

img = BytesIO()

plt.savefig(img,format='png',bbox\_inches='tight')

img.seek(0)

**return** base64.b64encode(img.getvalue())

In [3]:

**from** **pyecharts.charts** **import** WordCloud

**import** **wordcloud**

**from** **PIL** **import** Image

**import** **numpy** **as** **np**

**def** newPublisher(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

publisher = df[(**True**^df['出版社'].isin([0]))]

publisher['value'] = 1*#添加计数器*

df\_publisher = publisher[ ['出版社','value'] ].groupby(['出版社']).sum()*#按照出版社进行分组，得到不同出版社的图书数量*

df\_pbler=df\_publisher.sort\_values(by=['value'],ascending=**False**)

df\_pbler.to\_csv('出版社.csv')

wd = pd.read\_csv('出版社.csv')

word = [i[0] **for** i **in** wd[['出版社']].values]

value = [i[0] **for** i **in** wd[['value']].values]

*#print(word)*

*#print(value)*

words=""

index=0;

**for** i **in** word:

**if** word[index]!='\*\*\*':

index2=0

**for** index2 **in** range(value[index]):

words = words + word[index]+" "

index2=index2+1

*#print("words:"+words)*

index=index+1

m = np.array(Image.open("star.jpg"))

wc = wordcloud.WordCloud(

width=500,height=400,

*#设置词云输出图片的背景，字体步长，后面这个是随机种子，为了保证随机产生的结果一样，一般默认就是这样，不需要过多纠结*

background\_color='white',font\_step=3,random\_state=**False**,

*#词语水平方向排版出现的频率,默认 0.9 (所以词语垂直方向排版出现频率为 0.1 )*

mask= m,

prefer\_horizontal=0.8)

*#输出词云*

t=wc.generate(words)

t.to\_image().save('pub.png')

img = open('pub.png', 'rb').read()

**return** base64.b64encode(img)

**def** newWriter(path):

df = pd.read\_csv(path)

df.dropna()*#去掉空值*

author = df[(**True**^df['作者'].isin([0]))]

author['aut\_num'] = 1*#添加计数器*

df\_author = author[ ['作者','aut\_num'] ].groupby(['作者']).sum()*#按照作者进行分组，得到不同作者的图书数量*

df\_auth=df\_author.sort\_values(by=['aut\_num'],ascending=**False**)

df\_auth.to\_csv('作者.csv')

wr = pd.read\_csv('作者.csv')

auth = [i[0] **for** i **in** wr[['作者']].values]

fre = [i[0] **for** i **in** wr[['aut\_num']].values]

auths=""

index=0;

**for** i **in** auth:

index2=0

**for** index2 **in** range(fre[index]):

auths = auths + auth[index]+" "

index2=index2+1

index=index+1

m = np.array(Image.open("circle.jpg"))

wc = wordcloud.WordCloud(

width=500,height=400,

*#设置词云输出图片的背景，字体步长，后面这个是随机种子，为了保证随机产生的结果一样，一般默认就是这样，不需要过多纠结*

background\_color='gray',font\_step=3,random\_state=**False**,

*#词语水平方向排版出现的频率,默认 0.9 (所以词语垂直方向排版出现频率为 0.1 )*

mask= m,

prefer\_horizontal=0.8)

*#输出词云*

t=wc.generate(auths)

t.to\_image().save('aut.png')

img = open('aut.png', 'rb').read()

**return** base64.b64encode(img)

In [4]:

**from** **pyecharts.charts** **import** Tab

**import** **os**

**def** main():

path = "MLAB.csv"

*file = open(path, "w", encoding='utf-8')*

*List = getList()*

print("get list**\n**")

*listToCsv(List,file)*

print("store list**\n**")

ranks\_data = ranks(path)

print("ranks**\n**")

chigh\_data = comment\_high(path)

print("chigh**\n**")

auther\_data = auther(path)

print("auther**\n**")

cnum\_data = comment\_num(path)

print("cnum**\n**")

cnum2\_data = comment\_num2(path)

print("cnum2**\n**")

publisher = newPublisher(path)

print("publisher**\n**")

writer = newWriter(path)

print("writer**\n**")

*#二进制data集*

datas = [ranks\_data,publisher,writer,auther\_data,cnum\_data,cnum2\_data,chigh\_data]

root='''

<html>

<title>MLAB - data</title>

<body>

<div>

<span style='white-space:pre;'> </span>

'''

*#data转换为html*

index=1

**for** data **in** datas:

*#encoded = data*

index=index+1

**if** index==5:

root = root + '''

</div>

<div>'''

my\_html = '''

<img align = "middle" src="data:image/png;base64,**{}**" />

'''.format(data.decode('utf-8'))

imb = base64.b64encode(data)

root = root + my\_html

root = root + '''

</div>

</body>

</html>

'''

*#rtree解析字符串为html代码*

html = etree.HTML(root)

tree = etree.ElementTree(html)

tree.write('MLAB.html')

tab = Tab()

*# tab.add(Publisher,'Publisher')*

*# tab.add(Writer,'Writer')*

*#tab.render("MLAB.html")*

os.system("MLAB.html")

**if** \_\_name\_\_=='\_\_main\_\_':

main()