|  |  |  |
| --- | --- | --- |
| 姓名 | 学号 | 分工 |
|  |  | 组长 |
|  |  | 组员 |
|  |  | 组员 |
|  |  | 组员 |
|  |  | 组员 |

**爬虫实战项目**

|  |  |
| --- | --- |
| 报告题目 | 基于Python的豆瓣电影分析与可视化 |
| 完成日期 | 2019.6.17 |

目 录

1 需求分析-------------------------------------------------------------------------------------------3

* 1. 需求陈述-------------------------------------------------------------------------------------3

1. 设计阶段------------------------------------------------------------------------------------------4

2.1详细设计--------------------------------------------------------------------------------------------------4

3 代码实现--------------------------------------------------------------------------------------------5

3.1 爬虫部分-------------------------------------------------------------------------------------5

3.2 数据分析部分-------------------------------------------------------------------------------------------7

3.3 图表部分--------------------------------------------------------------------------------------10

1. 系统测试--------------------------------------------------------------------------------------------15

4.1 测试意义-------------------------------------------------------------------------------------15

4.2 测试流程---------------------------------------------------------------------------------------------------15

1. 项目运行结果&结果分析-------------------------------------------------------------------------15

5.1 电影分析部分-------------------------------------------------------------------------------15

5.2 用户推荐部分-------------------------------------------------------------------------------------------18

6 项目总结&感悟-------------------------------------------------------------------------------------19

6.1 项目总结-------------------------------------------------------------------------------------19

6.2 项目感悟-----------------------------------------------------------------19

## 1需求分析

1.1需求陈述

在这个豆瓣电影分析与分析系统中，我们使用了python语言来编写整个项目代码，分析所需数据完全利用爬虫爬取得到，爬取所得的数据被存入mysql数据库中，并提取出xls的数据被用作分析与可视化。

被爬取得到的数据被用于进行电影分析，我们将爬取到的top250部电影的信息按照影片发行国家和影片类型进行了分析并制作了图表来直观显示受欢迎的电影的分类。

## 2设计阶段

### 2.1 详细设计

在本阶段中，确定应该如何具体地实现所要求的网站信息系统，从而在编码阶段可以把这个描述直接翻译成用具体的 程序语言书写的程序。主要的工作有：根据在需求分析中所描述的数据、功能、运行，并依照概要设计所确定的处理流程、总体结构和模块外部设计，设计网站系统的结构、逐个模块的程序描述（包括各模块的功能、性能、输入、输出、算法、程序逻辑、接口等等），解决处理有关程序流程的问题 ：

1.接受：爬虫实现对所需信息的爬取；

2.输出：将爬取到的数据有序存入数据库；

3.信息过滤：对不符合爬取要求的数据进行处理，例如异常评分用户。

4.将数据作为参数传入两个处理模块；

5.在影片分析模块，分析数据并产生数据分析图表，包括扇形统计图，条形统计图和折线统计图；在用户推荐模块，利用协同过滤功能作出推荐列表，存入数据库并打印图表。

## 3代码实现

### 3.1 爬虫部分+数据库存取

### #导入包

### import pymysql

### import bs4

### import requests

### import re

### #连接数据库

### con =pymysql.connect(

### host="localhost",

### user="root",

### password="123456",

### db="movie",

### port=3306,

### use\_unicode=True,

### charset="utf8"

### )

### #获取游标

### cursor= con.cursor()

### #请求网页

### try:

### for i in range(0,50,25): #设置爬取的数量

### url = "https://movie.douban.com/top250?start={}&filter=".format(int(i))

### #设置请求头,如被封登录cookies来防止反爬

### headers = {

### 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36'

### }

### cookies = {

### 'cookie': 'bid=e0-x-JMRWJU; gr\_user\_id=bd38a54d-f6f6-4581-812f-54bee18e6e68; \_vwo\_uuid\_v2=D1F2B905DD84D187F0D39F9BE9CE73CCF|5eda4d43fa74ef681618d5014f9e22fc; douban-fav-remind=1; ll="108120"; viewed="30269348\_30233991\_3117898\_26801374\_3350010\_1200840"; \_\_utmc=30149280; \_\_utmc=223695111; push\_noty\_num=0; push\_doumail\_num=0; \_\_utmv=30149280.16039; \_\_gads=ID=e971665a65e6f278:T=1560449519:S=ALNI\_MZFS2WmZ5wrSiyS2cJmR9EfvjSDcQ; \_pk\_ref.100001.4cf6=%5B%22%22%2C%22%22%2C1560500086%2C%22https%3A%2F%2Fopen.weixin.qq.com%2Fconnect%2Fqrconnect%3Fappid%3Dwxd9c1c6bbd5d59980%26redirect\_uri%3Dhttps%253A%252F%252Fwww.douban.com%252Faccounts%252Fconnect%252Fwechat%252Fcallback%26response\_type%3Dcode%26scope%3Dsnsapi\_login%26state%3De0-x-JMRWJU%252523douban-web%252523https%25253A%252F%252Fmovie.douban.com%252Fsubject%252F1292052%252F%22%5D; \_pk\_ses.100001.4cf6=\*; \_\_utma=30149280.480877306.1535095302.1560452363.1560500086.28; \_\_utmz=30149280.1560500086.28.27.utmcsr=open.weixin.qq.com|utmccn=(referral)|utmcmd=referral|utmcct=/connect/qrconnect; \_\_utma=223695111.950917169.1560444124.1560453128.1560500086.4; \_\_utmb=223695111.0.10.1560500086; \_\_utmz=223695111.1560500086.4.3.utmcsr=open.weixin.qq.com|utmccn=(referral)|utmcmd=referral|utmcct=/connect/qrconnect; dbcl2="160397319:6ZpeKtCEwyQ"; ck=BBAp; douban-profile-remind=1; \_\_utmt=1; \_\_utmb=30149280.3.9.1560504224467; \_pk\_id.100001.4cf6=7991e268f9aa4410.1560444124.4.1560504237.1560453472.'

### }

### #请求URL

### req = requests.get(url, headers=headers, cookies=cookies)

### req.encoding='utf-8'

### #提取指定数据

### contents=req.text

### # print(contents)

### soup = bs4.BeautifulSoup(contents, "lxml")

### for tag in soup.find\_all('div',class\_='info'):

### #电影名称

### name = tag.find('span',class\_='title').get\_text()

### #评分

### rating\_score = float(tag.find('span', class\_='rating\_num').get\_text())

### people = tag.find('div', class\_="star")

### span = people.findAll('span')

### #评价

### peoplecount = span[3].contents[0]

### #URL

### url = tag.find('a').get('href')

### #进入URL提取内容

### req = requests.get(url)

### # print(req)

### req.encoding = "utf-8"

### contents = req.text

### soup = bs4.BeautifulSoup(contents, "html.parser")

### for tag2 in soup.find\_all('div', id='info'):

### m\_countrylist = re.findall("<span class=\"pl\">制片国家/地区:</span>(.\*)<br/>",str(tag2))

### m\_propertylist = re.findall("<span property=\"v:genre\">(.\*?)</span>",str(tag2))

### #提取内容

### countrylist = m\_countrylist[0].split('/')

### m\_country = ""

### for i in countrylist:

### j = i.strip()

### m\_country = m\_country + "," + j

### # 制片国家/地区

### m\_country = m\_country.strip(",")

### #类型

### m\_property = ','.join(m\_propertylist)

### #提取豆瓣评分

### for tag3 in soup.find\_all('div', class\_='rating\_wrap clearbox'):

### stars = re.findall("<span class=\"rating\_per\">(.\*)</span>",str(tag3))

### #五星

### five = stars[0]

### #四星

### four = stars[1]

### #三星

### three = stars[2]

### #两星

### two = stars[3]

### #一星

### one = stars[4]

### print("正在爬取："+name + " " + str(rating\_score) + " " + peoplecount + " " + url)

### sql = "insert into movie(电影名称,评分,评价,URL,制片国家,类型,五星,四星,三星,两星,一星) values("+"'"+name+ "'," + "'" +str(rating\_score)+ "',"+ "'" +peoplecount+ "'," + "'" +url+ "',"+ "'" +m\_country+"',"+ "'" +m\_property+ "',"+"'"+five+"',"+"'"+four+"',"+"'"+three+ "'," + "'" +two+ "'," + "'" +one+"'"")"

### cursor.execute(sql)

### con.commit()

### # 提取电影短评

### comment\_url = url + "/comments?status=P"

### req = requests.get(comment\_url)

### req.encoding = "utf-8"

### # print(req.text)

### contents = req.text

### #电影名

### movieName = re.findall("<title>(.\*) 短评</title>", str(contents))

### #评论人的id名

### username = re.findall("<a href=\"https://www.douban.com/people/(.\*)/\" class=\"\">", str(contents))

### #评分数

### stars = re.findall("<span class=\"allstar(.\*) rating", str(contents))

### # print(movieName,username,stars)

### sql = "select id from movie where 电影名称=" + "'" + movieName[0] + "'"

### # print(sql)

### cursor.execute(sql)

### #获取查询结果的第一行数据

### data = cursor.fetchone()

### con.commit()

### #插入短评

### movieId = str(data[0])

### for i in range(len(stars)):

### sql = "insert into commentDetail(电影id,评分数,评论人id) values(" + "'" + movieId + "'," + "'" + stars[i] + "'," + "'" + username[i] + "')"

### # print(sql)

### cursor.execute(sql)

### con.commit()

### except:

### cursor.execute("truncate table movie;truncate table commentdetail;")

### con.commit()

### print("出错了")

### 3.2数据分析（协同过滤）

### import pandas as pd

### import numpy as np

### import matplotlib.pyplot as plt

### #增加列头

### names = [

### 'id','电影名称','评分','评价','URL链接','制片国家','类型','五星','四星','三星','两星','一星'

### ]

### df = pd.read\_excel("movie.xls",header = None,names= names)

### #清除评价后面的汉字

### keywordlist = ['人评价']

### df['评价'] = df['评价'].str.replace('|'.join(keywordlist), ' ')

### # print(df)

### #统计制片国家数量

### production = df['制片国家'].str.split(',').apply(pd.Series)

### a = production.apply(pd.value\_counts).fillna('0')

### a.columns = ['area\_1','area\_2','area\_3','area\_4']

### a['area\_1'] = a['area\_1'].astype(int)

### a['area\_2'] = a['area\_2'].astype(int)

### a['area\_3'] = a['area\_3'].astype(int)

### a['area\_4'] = a['area\_4'].astype(int)

### a = a.apply(lambda x: x.sum(),axis = 1)

### number = pd.DataFrame(a, columns = ['数量'])

### # print(number.head(1000))

### #统计电影类型数量

### area\_split = df['类型'].str.split(',').apply(pd.Series)

### g = area\_split.apply(pd.value\_counts)

### g = g.unstack().dropna().reset\_index()

### g.columns = ['level\_0','类型', '数量']

### typenumber = g.drop(['level\_0'],axis = 1).groupby('类型').sum()

### # print(typenumber.head(1000))

### #评分排名

### score = df[['评分','电影名称']].sort\_values(by = ['评分'],ascending = False).head(100).reset\_index()

### # print(score)

### #评价排名

### evaluate = df[['评价','电影名称']].sort\_values(by = ['评价'],ascending = False).head(100).reset\_index()

### # print(evaluate)

### # 总排名

### # print(df[['id','电影名称']].head(100))

### writer = pd.ExcelWriter('清洗后数据.xlsx')

### number.to\_excel(writer, sheet\_name = '电影制片国家数量统计', index = True)

### typenumber.to\_excel(writer, sheet\_name = '电影类型数量总数统计', index = True)

### score.to\_excel(writer, sheet\_name = '评分排名统计', index = False)

### evaluate.to\_excel(writer, sheet\_name = '评价排名统计', index = False)

### df[['id','电影名称']].to\_excel(writer, sheet\_name = '总排名', index = False)

### print('保存完成')

### writer.save()

### writer.close()

### 3

### .3 图表部分

from pyecharts import \*

import xlrd

#打开数据文件

workbook = xlrd.open\_workbook('清洗后数据.xlsx')

# 获取workbook中所有的表格

sheet1 = workbook.sheet\_by\_index(0)

sheet2 = workbook.sheet\_by\_index(1)

sheet3 = workbook.sheet\_by\_index(2)

sheet4 = workbook.sheet\_by\_index(3)

# 打印所有表

# print(workbook.sheet\_names())

# 获取第一个表的数据

a =sheet1.col\_values(0)[1:]

b =sheet1.col\_values(1)[1:]

# 获取第二个表的数据

a1 =sheet2.col\_values(0)[1:]

b1 =sheet2.col\_values(1)[1:]

# 获取第三个表的数据

a2 =sheet3.col\_values(1)[1:]

b2 =sheet3.col\_values(2)[1:]

# 获取第四个表的数据

a3 =sheet4.col\_values(1)[1:]

b3 =sheet4.col\_values(2)[1:]

# 实例化page类

page = Page()

# Bar

bar=Bar("柱状图示例")

bar.add("数量",a,b,is\_label\_show=True,xaxis\_interval=0,is\_datazoom\_show=True,datazoom\_type='both',mark\_point=["max", "min"])

page.add(bar)

bar1 = Bar("")

bar1.add("分数", b2, a2,is\_label\_show=True,is\_datazoom\_show=True, is\_stack=True)

page.add(bar1) # TODO 向page中添加图表

# Funnel

funnel=Funnel("漏斗图示例")

funnel.add('占比',a1,b1)

page.add(funnel)# TODO 向page中添加图表

# pie

pie = Pie("饼图-圆环图示例", title\_pos='center')

pie.add("比例", a1, b1, radius=[40, 75], label\_text\_color=None,is\_label\_show=True, legend\_orient='vertical', legend\_pos='left')

page.add(pie) # TODO 向page中添加图表

# line

line = Line("折线图示例")

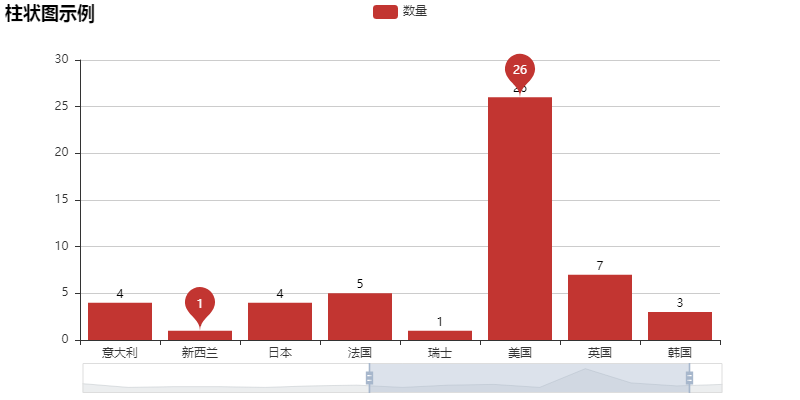
line.add("评价数量",b3,a3 ,mark\_point=["max", "min"], is\_label\_show=True,is\_datazoom\_show=True)

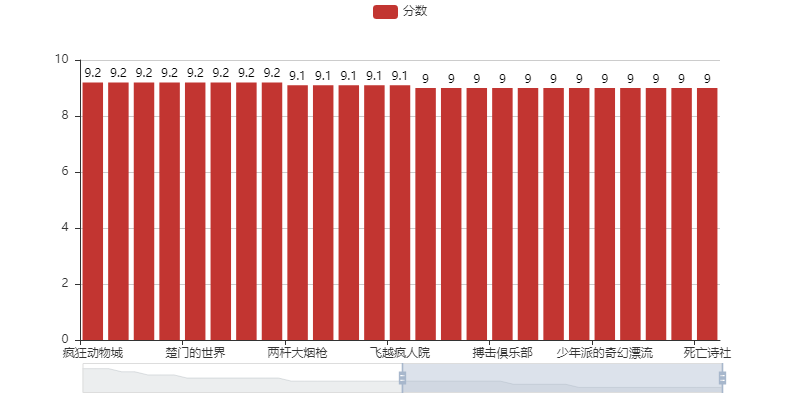
page.add(line)# TODO 向page中添加图表

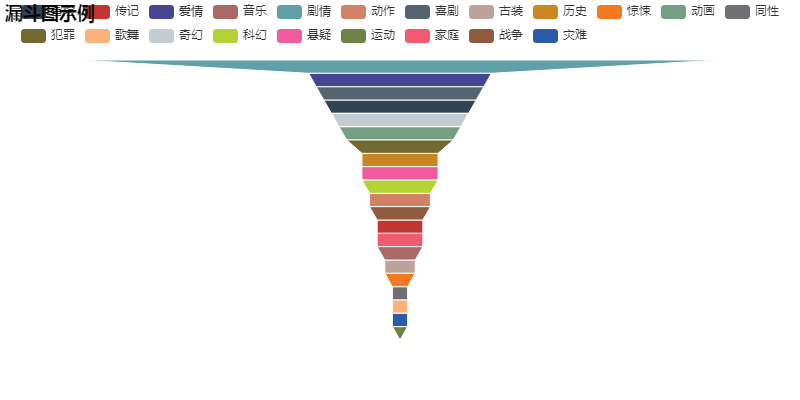
page.render("可视化图.html")

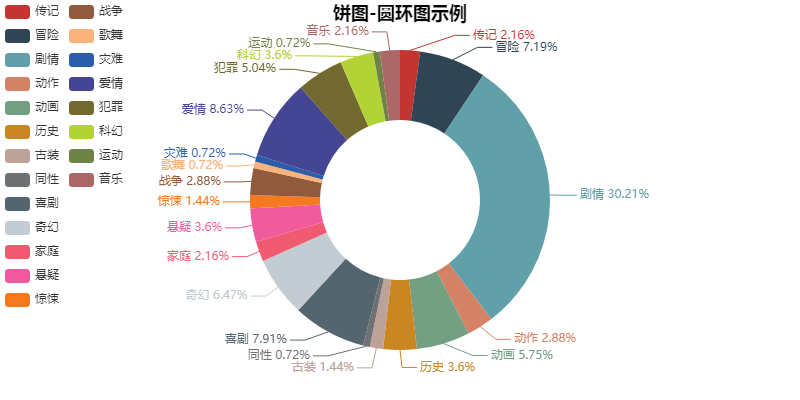
## 5 项目运行结果&结果分析

### 5.1电影分析部分











## 项目总结

通过本次项目的学习，我们领会到了python语言的魅力，以及小组协作学习的乐趣。我们基于当下电影市场的火热情况选定了电影分析这个项目主题。做项目的时候，最重要的不是自己如何快速地将自己分配的任务做完，而是要注重团队配合。一开始组内必须对这个项目的数据库的命名进行讨论，定义表的属性的数据类型，表与表之间会有关联，所以有的属性的类型与长度必须定义一致，这样访问的数据库才不会出错误，如果一开始不将这些步骤统一下来的话，就会给后面的编程带来一系列的问题。

当项目组长将这个项目组合完成之后，我知道了做一个项目不是一个人就能够完成的，需要组员之间的配合，因为有的功能与别人做的功能是相连接的。如果只顾自己，这个项目也不会完成。这需要与组内的成员进行交流，以及组长的辛苦努力与组员们一起解决问题。