|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | Github账号 | 分工 | 成绩 |
| 蒋筱斌 | 201592268 | coding-j | 组长 |  |
| 郗晓斌 | 201592422 | 18742021291 | 组员 |  |
| 吕晓特 | 201592408 | lxt006 | 组员 |  |
| 谢京达 | 201592464 | 645503801@qq.com | 组员 |  |
| 张胜仁 | 201592349 | 18742013308 | 组员 |  |

**开源软件基础课程报告**

|  |  |
| --- | --- |
| 报告题目 | 基于Python的豆瓣电影分析与推荐 |
| 项目网址 | <https://github.com/coding-j/movieAnalysis> |
| 完成日期 | 2018.1.11 |

大连理工大学软件学院

目 录

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

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

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

2.1概要设计-------------------------------------------------------------------------------------4

2.2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

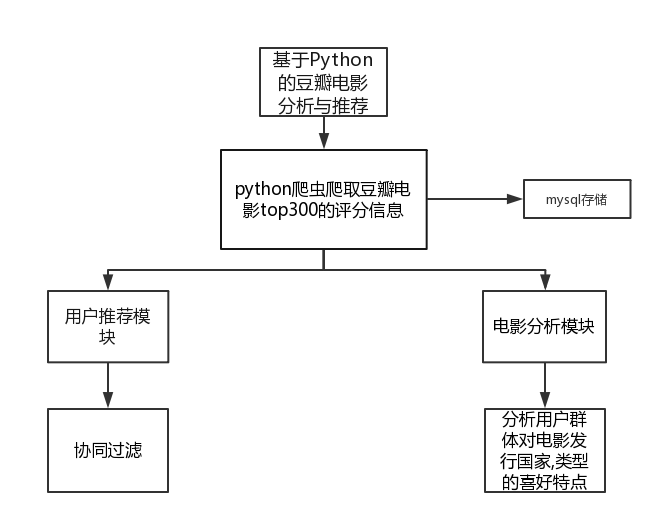
在这个豆瓣电影分析与推荐系统中，我们使用了PYTHON语言来编写整个项目代码，分析所需数据完全利用爬虫爬取得到，爬取所得的数据被存入MYSQL数据库中，取得的数据被用作两个方面。

其一，被爬取得到的数据被用于进行电影分析，我们将爬取到的top300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。

其二，我们利用了一种用于推荐系统的算法---协同过滤算法，来为每位参与电影评分的用户筛选出推荐电影内容，首先读入数据，形成用户-电影矩阵，其次根据用户-电影矩阵计算不同电影之间的相关系数（一般用person相关系数），形成电影-电影相关度矩阵。然后根据电影-电影相关度矩阵，以及用户已有的评分，通过加权平均计算用户未评分电影的预估评分。例如用户对A电影评3分、B电影评4分、C电影未评分，而C电影与A电影、B电影的相关度分别为0.3和0.8，则C电影的预估评分为(0.3\*3+0.8\*4)/(0.3+0.8)。

## 2设计阶段

2.1 概要设计

项目功能流程图

### 2.2 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

import pymysql

import bs4

import requests

import re

con = pymysql.connect(

host="localhost",

user="root",

password="123456",

db="movie",

port=3306,

use\_unicode=True,

charset="utf8"

)

cursor = con.cursor()

try:

for i in range(0, 25, 25):

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

# print(url)

req = requests.get(url)

req.encoding="utf-8"

#print(req.text)

contents = req.text

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

# print("豆瓣电影TOP250" + "\n" +" 影片名 评分 评价人数 链接 ")

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

m\_name = tag.find('span',class\_='title').get\_text()

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

m\_people = tag.find('div', class\_="star")

m\_span = m\_people.findAll('span')

m\_peoplecount = m\_span[3].contents[0]

m\_url = tag.find('a').get('href')

if m\_url == "https://movie.douban.com/subject/5912992/":

continue

req = requests.get(m\_url)

req.encoding = "utf-8"

# print(req.text)

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))

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

m\_country = ""

for i in countrylist:

j = i.strip()

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

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

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

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

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

five = stars[0]

four = stars[1]

three = stars[2]

two = stars[3]

one = stars[4]

print(m\_name + " " + str(m\_rating\_score) + " " + m\_peoplecount + " " + m\_url)

sql = "insert into movie(name,score,commentNum,link,country,property,five\_star,four\_star,three\_star,two\_star,one\_star) values('"+m\_name+"',"+str(m\_rating\_score)+",'"+m\_peoplecount+"','"+m\_url+"','"+m\_country+"','"+m\_property+"','"+five+"','"+four+"','"+three+"','"+two+"','"+one+"')"

cursor.execute(sql)

con.commit()

comment\_url = m\_url + "/comments?status=P"

req = requests.get(comment\_url)

req.encoding = "utf-8"

# print(req.text)

contents = req.text

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

# print(movieName)

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

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

sql = "select Id from movie where name=" + "'" + movieName[0] + "'"

# print(sql)

cursor.execute(sql)

data = cursor.fetchone()

con.commit()

movieId = str(data[0])

for i in range(len(stars)):

sql = "insert into commentDetail(movieId,stars,username) values(" + "'" + movieId + "'," + "'" + stars[i] + "'," + "'" + username[i] + "')"

print(sql)

cursor.execute(sql)

con.commit()

# except Exception as e:

# con.rollback()

finally:

con.close()

### 3.2数据分析（协同过滤）

import sys

import numpy as np

import pandas as pd

from numpy import linalg as la

import csv

username = list(range(313))

data = {}

f = open("user.txt")

line = f.readline()

i=0

while line:

username[i] = line.rstrip("\n")

line = f.readline()

i=i+1

f.close()

print(username)

f = open("result.txt",encoding='UTF-8')

line = f.readline()

while line:

a = line.split("\t")

# print(a[0],a[1],a[2])

# print(type(a[2]))

if a[0] not in data:

data[a[0]] = {}

data[a[0]][a[1]] = int(a[2])

else:

if a[1] not in data[a[0]]:

data[a[0]][a[1]] = int(a[2])

else:

data[a[0]][a[1]].append(int(a[2]))

line = f.readline()

f.close()

print(data)

# clean&transform the data

data = pd.DataFrame(data)

# 0 represents not been rated

data = data.fillna(0)

# each column represents a movie

mdata = data.T

# calculate the simularity of different movies, normalize the data into [0,1]

np.set\_printoptions(3)

mcors = np.corrcoef(mdata, rowvar=0)

mcors = 0.5 + mcors \* 0.5

mcors = pd.DataFrame(mcors, columns=mdata.columns, index=mdata.columns)

# calculate the score of every item of every user

# matrix:the user-movie matrix

# mcors:the movie-movie correlation matrix

# item:the movie id

# user:the user id

# score:score of movie for the specific user

def cal\_score(matrix, mcors, item, user):

totscore = 0

totsims = 0

score = 0

if pd.isnull(matrix[item][user]) or matrix[item][user] == 0:

for mitem in matrix.columns:

if matrix[mitem][user] == 0:

continue

else:

totscore += matrix[mitem][user] \* mcors[item][mitem]

totsims += mcors[item][mitem]

score = totscore / totsims

else:

score = matrix[item][user]

return score

# calculate the socre matrix

# matrix:the user-movie matrix

# mcors:the movie-movie correlation matrix

# score\_matrix:score matrix of movie for different users

def cal\_matscore(matrix, mcors):

score\_matrix = np.zeros(matrix.shape)

score\_matrix = pd.DataFrame(score\_matrix, columns=matrix.columns, index=matrix.index)

for mitem in score\_matrix.columns:

for muser in score\_matrix.index:

score\_matrix[mitem][muser] = cal\_score(matrix, mcors, mitem, muser)

return score\_matrix

# give recommendations: depending on the score matrix

# matrix:the user-movie matrix

# score\_matrix:score matrix of movie for different users

# user:the user id

# n:the number of recommendations

def recommend(matrix, score\_matrix, user, n):

user\_ratings = matrix.ix[user]

not\_rated\_item = user\_ratings[user\_ratings == 0]

recom\_items = {}

# recom\_items={'a':1,'b':7,'c':3}

for item in not\_rated\_item.index:

recom\_items[item] = score\_matrix[item][user]

recom\_items = pd.Series(recom\_items)

recom\_items = recom\_items.sort\_values(ascending=False)

return recom\_items[:n]

# main

score\_matrix = cal\_matscore(mdata, mcors)

with open("re\_result.csv","w") as f:

writer = csv.writer(f)

for i in username:

user = i

print(user)

print(recommend(mdata, score\_matrix, user, 2))

# s = recommend(mdata, score\_matrix, user, 2)

# s.index.name = user

# s.to\_csv("test.csv")

# writer.writerows(s)

print("end")

### 3.3 图表部分

# -\*- coding: UTF-8 -\*-

import requests

import bs4

import pymysql

import re

import matplotlib.pyplot as plt

import matplotlib

from wordcloud import WordCloud,STOPWORDS,ImageColorGenerator

import jieba

import codecs

import os

from os import path

from scipy.misc import imread

con = pymysql.connect(

host="localhost",

user="root",

password="123456",

db="movie",

port=3306,

use\_unicode=True,

charset="utf8"

)

cursor = con.cursor()

plt.figure(1)

plt.figure(2)

country = {}

property = {}

try:

# 制片国家

sql = "select country from movie"

cursor.execute(sql)

data = cursor.fetchall()

con.commit()

for i in range(len(data)):

for j in range(len(data[i])):

m\_country = data[i][j].split(',')

for x in m\_country:

if x in country.keys():

country[x] = country[x] + 1

else:

country[x] = 1

# 电影类型

sql = "select property from movie"

cursor.execute(sql)

data = cursor.fetchall()

con.commit()

for i in range(len(data)):

for j in range(len(data[i])):

m\_property = data[i][j].split(',')

for x in m\_property:

if x in property.keys():

property[x] = property[x] + 1

else:

property[x] = 1

#国家的柱状图

# print(country.keys())

# print(country.values())

# print(country)

x = country.keys()

y = country.values()

# plt.figure(1)

plt.subplot(321)

#国家分类

#条形图

plt.ylabel("数量") #y lable

a=[1,2,3,4,5,6,7,8,9,10,11,12]

# y=[12, 2, 3, 4, 3, 2, 3, 1, 1, 1, 1, 1]

plt.bar(a,y,color='r')

labels=x

plt.xticks(a,labels, rotation=0)

plt.title('影片国家分布') #title

plt.subplot(325)

#饼形图

slices = y #number

activities = x

cols = ['c','m','r','b','lawngreen','darkred','m','deeppink','navy','grey','peru','olive']

plt.pie(slices,

labels=activities,

colors=cols,

startangle=90,

)

#折线图

plt.subplot(323)

plt.ylabel("数量") #y lable

a=[1,2,3,4,5,6,7,8,9,10,11,12]

# y=[12, 2, 3, 4, 3, 2, 3, 1, 1, 1, 1, 1]

plt.plot(a,y,color='r')

labels=x

plt.xticks(a,labels, rotation=0)

m=property.keys()

n=property.values()

#类型分类

#条形图

plt.subplot(322)

plt.ylabel("数量") #y lable

b=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]

plt.bar(b,n,color='b')

labels=m

plt.xticks(b,labels, rotation=0)

plt.title('影片类型分布') #title

plt.subplot(326)

#饼形图

slices = n #number

activities =m

cols = ['c','m','r','b','lawngreen','darkred','m','deeppink','navy','grey','peru','olive','palegreen']

plt.pie(slices,

labels=activities,

colors=cols,

startangle=90,

)

#折线图

plt.subplot(324)

plt.ylabel("数量") #y lable

b=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]

plt.plot(b,n,color='b')

labels=m

plt.xticks(b,labels, rotation=0)

plt.show()

#类型词云

d = path.dirname(\_\_file\_\_)

alice\_coloring = imread(path.join(d,"1.png"))

print("1")

wc = WordCloud(

background\_color="white",

mask=alice\_coloring,

font\_path='C:\Windows\Fonts\STZHONGS.TTF',

stopwords=STOPWORDS,

max\_font\_size=500,

random\_state=42

)

print(property)

wc.generate\_from\_frequencies(property)

image\_colors = ImageColorGenerator(alice\_coloring)

# 以下代码显示图片

# plt.figure()

plt.imshow(wc)

plt.axis("off")

plt.show()

# except Exception as e:

# con.rollback()

finally:

con.close()

## 4 系统测试

4.1测试意义

软件的测试在软件生命周期中占据重要的地位，在软件生命周期每一阶段中都应包含测试，从而检验本阶段的成果是否接近预期的目标，尽可能早的发现错误并加以修正，如果不在早期阶段进行测试，错误的延时扩散常常会导致最后成品测试的巨大困难，

### 4.2测试流程

与开发过程类似，测试过程也必须分步骤进行，每个步骤在逻辑上是前一个步骤的继续。测试基本上由下述几个步骤组成：

1. 模块测试 对概要设计和模块设计的结构进行分析与测试，以避免初期返工。

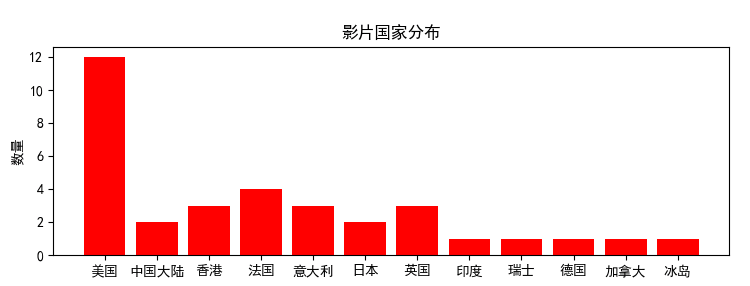
2. 系统测试 测试整个系统的稳定性和系统速度。

3. 验收测试 对最后的图表进行直观分析是否和实际存在巨大误差。

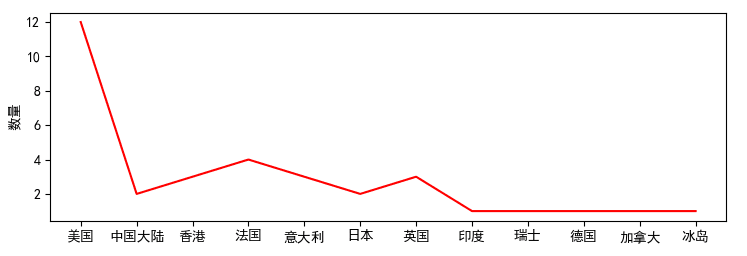
## 5 项目运行结果&结果分析

### 5.1电影分析部分

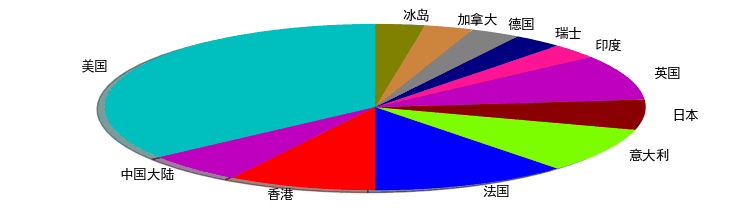
影片国家条形分布图



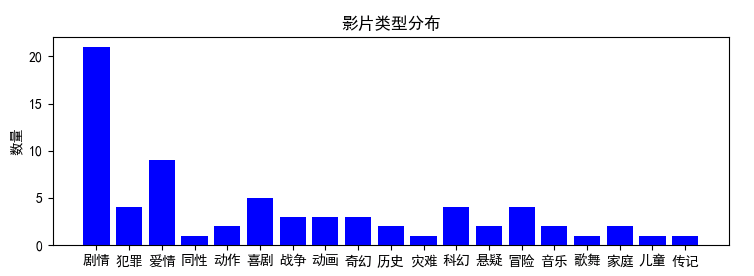
影片国家折线分布图



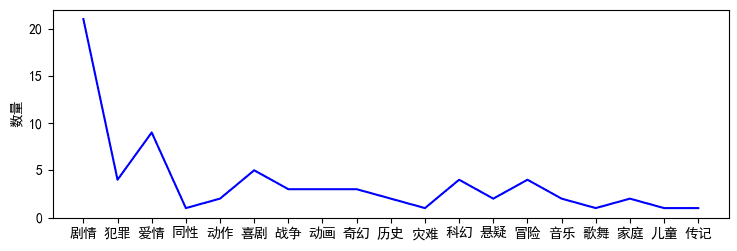
影片国家扇形分布图

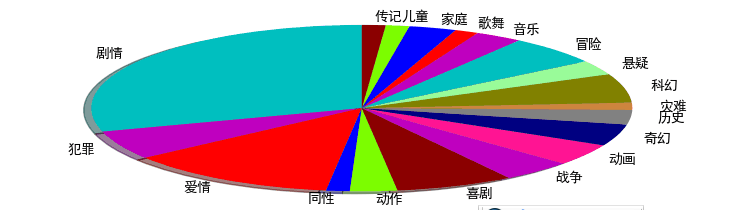


影片类型条形分布图



影片类型折线分布图

 影片类型扇形分布图



### 5.2 用户推荐部分

这个词云是根据用户的评分情况计算得到的词云图，图中的词语大小代表该类影片受到用户的欢迎程度，欢迎程度越高，词语越大





上面是为每位用户推荐两部影片的部分记录,权值代表每部电影的推荐优先度.

## 项目总结&感悟

### 6.1项目总结

在这个豆瓣电影分析与推荐系统中，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，取得的数据被用作两个方面。

其一，被爬取得到的数据被用于进行电影分析，我们将爬取到的top300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。

其二，我们利用了一种用于推荐系统的算法---协同过滤算法，来为每位参与电影评分的用户筛选出推荐电影内容，首先读入数据，形成用户-电影矩阵，其次根据用户-电影矩阵计算不同电影之间的相关系数（一般用person相关系数），形成电影-电影相关度矩阵。然后根据电影-电影相关度矩阵，以及用户已有的评分，通过加权平均计算用户未评分电影的预估评分。例如用户对A电影评3分、B电影评4分、C电影未评分，而C电影与A电影、B电影的相关度分别为0.3和0.8，则C电影的预估评分为(0.3\*3+0.8\*4)/(0.3+0.8)。

然后，我们实现了一些创新的功能。首先，我们爬取网页当中电影的数据，对网页当中的电影数据进行分析，分类，确定不同类型电影的百分比。然后我们分析用户对不同类型电影的打分情况，运用同类分析的方法对用户进行推荐其可能感兴趣的电影。

### 6.2 项目感悟

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。