****

基于协同过滤的网易云音乐

数据统计与推荐系统

一组 | 数据分析与处理软件开发综合实践 | 2018.11

小组成员：

李琦 岳名扬 袁名扬 杜玉楷 黎政君 何士豪 鲁芯丝雨 杨畅

目录

[一、 前言 3](#_Toc531278189)

[1.1 实践目的 3](#_Toc531278190)

[1.2 实践要求 3](#_Toc531278191)

[1.3 实践背景 3](#_Toc531278192)

[1.4 实践意义 3](#_Toc531278193)

[二、 系统概要 4](#_Toc531278194)

[2.1 设计思路 4](#_Toc531278195)

[2.2 系统组成 4](#_Toc531278196)

[2.2.1 系统框图 4](#_Toc531278197)

[2.2.2 系统流程图 5](#_Toc531278198)

[2.2.3 系统用例图 6](#_Toc531278199)

[2.3 系统功能 6](#_Toc531278200)

[2.3.1 数据可视化 6](#_Toc531278201)

[2.3.2 个性化推荐 6](#_Toc531278202)

[三、 小组分工完成情况及个人负责代码 6](#_Toc531278203)

[3.1 分工情况 6](#_Toc531278204)

[3.2 数据库设计 7](#_Toc531278205)

[3.2.1 设计思路 7](#_Toc531278206)

[3.2.2 代码展示 9](#_Toc531278207)

[3.3 数据获取与清洗 15](#_Toc531278208)

[3.3.1 设计思路 15](#_Toc531278209)

[3.3.2 代码展示 15](#_Toc531278210)

[3.4 推荐算法应用 17](#_Toc531278211)

[3.4.1 设计思路 17](#_Toc531278212)

[3.4.2 代码展示 17](#_Toc531278213)

[3.5 界面显示与数据可视化 23](#_Toc531278214)

[3.5.1 设计思路 23](#_Toc531278215)

[3.5.2 代码展示 26](#_Toc531278216)

[四、 项目总结 33](#_Toc531278217)

[1. 岳名扬 33](#_Toc531278218)

[2. 袁名扬 33](#_Toc531278219)

[3. 杜玉楷 33](#_Toc531278220)

[4. 李琦 34](#_Toc531278221)

[5. 黎政君 34](#_Toc531278222)

[6. 何士豪 34](#_Toc531278223)

[7. 鲁芯丝雨 35](#_Toc531278224)

[8. 杨畅 35](#_Toc531278225)

[五、 组员成绩表 35](#_Toc531278226)

[六、 参考文献 36](#_Toc531278227)

# 前言

## 实践目的

　　为了将理论用于实践，巩固所学知识，提高自己发现问题并用所学知识分析问题和解决问题的能力，锻炼自己的工作能力，适应社会能力，自我管理能力，了解目前软件的应用情况，需求情况，发展方向及前景，为顺利毕业做好充分的准备，也为自己能顺利的与社会环境接轨做好准。.此次综合实践由学校统一组织安排，分四个方向进行系统的综合实践。

## 实践要求

综合实践要求：

（1）要求学生在实践过程中认真学习技术知识，积极与指导老师和同学配合； （2）在前期，按时到勤，认真学习。对技术的理论知识要及时实践；

（3）在后期，积极与同学沟通，认真完成项目要求的内容。在这个过程中要与老师同学多做沟通，通过探讨项目的解决方案以及进展；（4）实践时间：2018年11月5日至2018年11月30日。

## 实践背景

　　当前时代是一个信息爆炸的时代，数据成了当下经济的代名词。虽然数据里边蕴含着各种各样的需求，发展方向，但是庞杂的数据不经过分析处理是无法被正确良好的运用。因此我们准备运用爬虫技术，分析数据，从而找到用户的潜在音乐需求，带给用户更好的音乐体验。

## 实践意义

　　通过项目开发的形式，让同学们了解到所学知识如何与现实生活需要相对接：一方面它让同学们了解了现实生活中的情况；另一方面，它也锻炼了同学们的实践能力，拓宽了学生的眼界。主要有以下几个方面的好处：

（1）培养了学生的实践能力，复习了之前所学的知识，并且在此基础之上，探索新的知识，培养了学生的自学习能力；

（2）有利于培养学生日后工作能力；

（3）锻炼了学生的编程能力；（4）有利于同学们对于项目开发流程的理解。

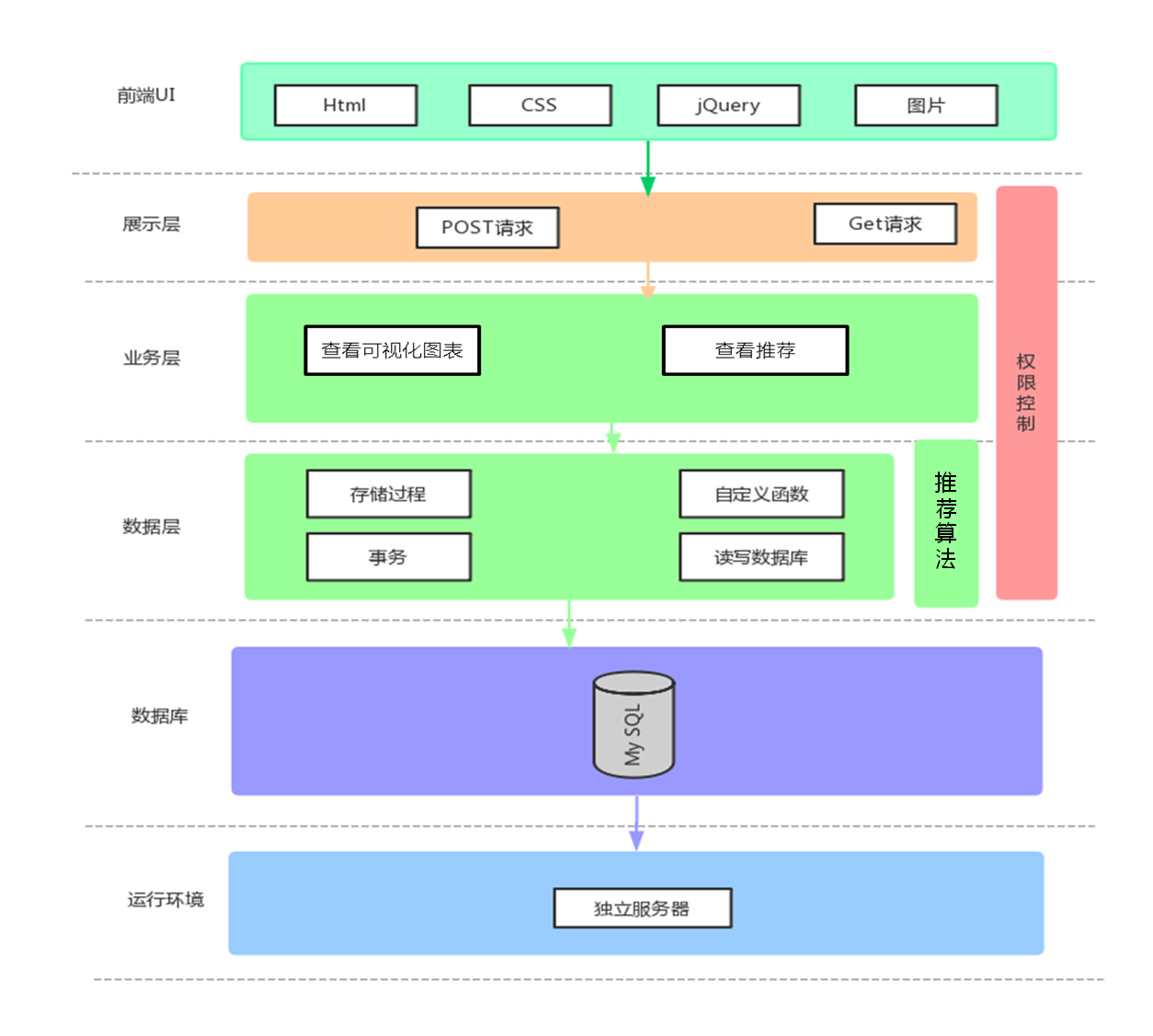
# 系统概要

## 设计思路

　　系统整体分为前端和后台两个部分，其中前端的主要功能是进行数据可视化以及推荐结果的显示；后台的主要功能是通过爬虫获取网页数据经清洗和整理后存入数据库备用、前端提供数据交换接口以及进行推荐算法应用。

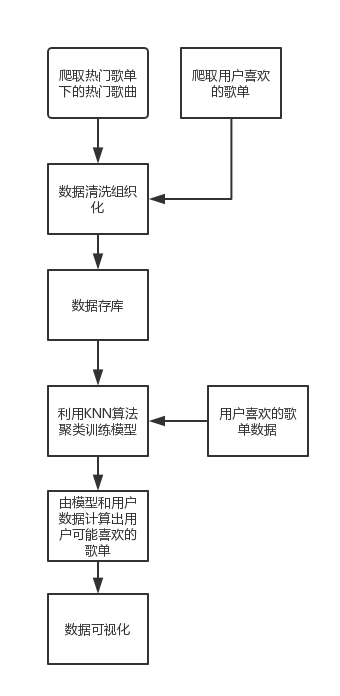
## 系统组成

### 系统框图



　　系统结构框图：主要包括管理系统实现的整体逻辑过程与部分功能，如整体分为五层：前端UI、展示层、业务层、数据层和数据库。我们系统主要基于Python语言，使用的Python框架包括爬虫框架Scrapy、推荐系统框架surprise、web框架Flask等，此外在可视化方面使用了百度开源ECharts库，采用的数据库是MySQL。

### 系统流程图



　　首先使用爬虫模块从网易云音乐服务器获取数据，之后将其清洗、整理并存入数据库。前端调取数据库内数据进行图表绘制实现数据可视化，并在训练好推荐模型的情况下向用户进行歌单或歌曲的推荐。

### 系统用例图



## 系统功能

用户进入系统后，可查看图表化后的统计信息，并查看结合自己的行为历史而得出的个性化推荐结果。

### 数据可视化

用户可以查看的统计信息包括：

(1) 热门歌曲的评论词云与评论随时间的数量分布，歌曲评论的被点赞数分布；

(2) 热门歌单的标签词云；

(3) 艺人或歌曲的受欢迎程度，受欢迎程度以歌曲或歌手在用户的听歌排行前100名内次数的多寡以及在听歌排行中的名次作为综合的衡量标准；

(4) 查看喜好相投的用户的年龄段和地域分布。

### 个性化推荐

　　用户可以根据自己的行为历史获得可能喜欢的歌曲或者歌单的推荐。

# 小组分工完成情况及个人负责代码

## 分工情况

何士豪、岳名扬：数据爬取，推荐算法；

鲁芯丝雨、袁名扬、黎政君、杜玉楷：数据可视化；

李琦、杨畅：数据库设计。

## 数据库设计

### 设计思路

　　经分析，我们设计的数据库中共有6个实体，分别是歌曲、歌单、艺人、用户、评论和标签，主要关系共有7种。E-R图如下：



与之对应的数据字典如下：

1. 用户 Uuser

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| UserID | varchar(255) |  | yes | 主键 | 用户ID |
| UserName | varchar(255) |  | yes |  | 用户名 |
| UserBirth | int |  | yes |  | 用户生日 |
| UserArea | varchar(255) |  | yes |  | 用户所在地 |

2. 艺人 Artist

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| ArtistID | varchar(255) |  | yes | 主键 | 艺人ID |
| ArtistName | varchar(255) |  | yes |  | 艺人名 |

3. 歌曲Song

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| SongID | varchar(255) |  | yes | 主键 | 歌曲ID |
| SongName | varchar(255) |  | yes |  | 歌曲名 |
| PlayTime | int |  | yes |  | 播放时长 |
| PubTime | int |  | yes |  | 发布时间 |

4. 歌单Playlist

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| PlaylistID | varchar(255) |  | yes | 主键 | 歌单ID |
| PlaylistName | varchar(255) |  | yes |  | 歌单名 |
| UserID | varchar(255) |  | yes | Uuser表外键 | 创建用户ID |

5. 评论Comment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| CommentID | varchar(255) |  | yes | 主键 | 评论ID |
| CommentTime | int |  | yes |  | 评论时间 |
| LikedCount | int |  | yes |  | 被点赞数 |
| CommentContent | varchar(255) |  | yes |  | 歌单名 |
| UserID | varchar(255) |  | yes | Uuser表外键 | 用户ID |
| SongID | varchar(255) |  | yes | Song表外键 | 歌曲ID |

6. 艺人表演歌曲 Artist\_Song

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| ArtistID | varchar(255) |  | yes | 主键，Artist表外键 | 艺人ID |
| SongID | varchar(255) |  | yes | 主键，Song表外键 | 歌曲ID |

7. 歌单包含标签 Playlist\_Tag

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| PlaylistID | varchar(255) |  | yes | 主键，Playlist表外键 | 歌单ID |
| TagName | varchar(255) |  | yes | 主键 | 标签名 |

8. 歌曲属于歌单 Playlist\_Song

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| PlaylistID | varchar(255) |  | yes | 主键，Playlist表外键 | 歌单ID |
| SongID | varchar(255) |  | yes | 主键，Song表外键 | 歌曲ID |

9. 用户喜欢歌曲 User\_Song

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 默认值 | 非空 | 键 | 备注 |
| UserID | varchar(255) |  | yes | 主键，Uuser表外键 | 用户ID |
| SongID | varchar(255) |  | yes | 主键，Song表外键 | 歌曲ID |
| SongScore | int |  | yes |  | 听歌频率占比 |

基于前端所需功能，我们也对SQL建表以及数据库操作语句进行了组织。

### 代码展示

建表SQL语句如下：

CREATE DATABASE `music`

CREATE TABLE `uuser` (

`UserID` varchar(255) NOT NULL ,

`UserName` varchar(255) NOT NULL ,

`UserBirth` int NOT NULL ,

`UserArea` varchar(255) NOT NULL ,

PRIMARY KEY (`UserID`)

)

CREATE TABLE `artist` (

`ArtistID` varchar(255) NOT NULL ,

`ArtistName` varchar(255) NOT NULL ,

PRIMARY KEY (`ArtistID`)

)

CREATE TABLE `song` (

`SongID` varchar(255) NOT NULL ,

`SongName` varchar(255) NOT NULL ,

`PlayTime` int NOT NULL ,

`PubTime` int NOT NULL ,

PRIMARY KEY (`SongID`)

)

CREATE TABLE `playlist` (

`PlaylistID` varchar(255) NOT NULL ,

`PlaylistName` varchar(255) NOT NULL ,

`UserID` varchar(255) NOT NULL ,

PRIMARY KEY (`PlaylistID`) ,

FOREIGN KEY (`UserID`) REFERENCES `uuser` (`UserID`)

)

CREATE TABLE `comment` (

`CommentID` varchar(255) NOT NULL ,

`CommentTime` int NOT NULL ,

`LikedCount` int NOT NULL ,

`CommentContent` varchar(255) NOT NULL ,

`UserID` varchar(255) NOT NULL ,

`SongID` varchar(255) NOT NULL ,

PRIMARY KEY (`CommentID`) ,

FOREIGN KEY (`UserID`) REFERENCES `uuser` (`UserID`) ,

FOREIGN KEY (`SongID`) REFERENCES `song` (`SongID`)

)

CREATE TABLE `artist\_song` (

`ArtistID` varchar(255) NOT NULL ,

`SongID` varchar(255) NOT NULL ,

PRIMARY KEY (`ArtistID`,`SongID`) ,

FOREIGN KEY (`ArtistID`) REFERENCES `artist` (`ArtistID`) ,

FOREIGN KEY (`SongID`) REFERENCES `song` (`SongID`)

)

CREATE TABLE `playlist\_tag` (

`PlaylistID` varchar(255) NOT NULL ,

`TagName` varchar(255) NOT NULL ,

PRIMARY KEY (`PlaylistID`,`TagName`) ,

FOREIGN KEY (`PlaylistID`) REFERENCES `playlist` (`PlaylistID`)

)

CREATE TABLE `playlist\_song` (

`PlaylistID` varchar(255) NOT NULL ,

`SongID` varchar(255) NOT NULL ,

PRIMARY KEY (`PlaylistID`,`SongID`) ,

FOREIGN KEY (`PlaylistID`) REFERENCES `playlist` (`PlaylistID`) ,

FOREIGN KEY (`SongID`) REFERENCES `song` (`SongID`)

)

CREATE TABLE `user\_song` (

`UserID` varchar(255) NOT NULL ,

`SongID` varchar(255) NOT NULL ,

`SongScore` int NOT NULL ,

PRIMARY KEY (`UserID`,`SongID`) ,

FOREIGN KEY (`UserID`) REFERENCES `uuser` (`UserID`) ,

FOREIGN KEY (`SongID`) REFERENCES `song` (`SongID`)

)

SQL查询语句如下：

SELECT COUNT(\*) FROM artist;

SELECT COUNT(\*) FROM song WHERE PubTime IS NOT NULL;

SELECT COUNT(\*) FROM SongHasArtist;

SELECT COUNT(\*) FROM user\_score;

SELECT \* FROM uuser WHERE userarea is NOT NULL

SELECT \* FROM song WHERE playtime >300000

SELECT \* FROM `comment` WHERE `comment`.commentId='382983247'

SELECT uuser.userId,COUNT(\*) as total FROM user\_score,uuser WHERE uuser.UserID=user\_score.userId AND uuser.flag!='NoPermission' GROUP BY UserID ORDER BY COUNT(\*) DESC

SELECT \* FROM uuser WHERE username='网易云音乐'

SELECT songId,COUNT(\*) as total FROM song\_comment GROUP BY songId ORDER BY COUNT(\*) #DESC\*

SELECT \* from `comment` WHERE `comment`.likedCount<100

SELECT songid FROM user\_score WHERE songid IN(SELECT songid FROM song)

UPDATE uuser SET flag='Default'

UPDATE set playTime=playTime/1000 from song where playTime IS NOT NULL ORDER BY playTime DESC

(SELECT song.songId,userid,song.SongName FROM user\_score,song where song.SongID=user\_score.songid and song.songid not in(select song.SongID FROM song WHERE playTime is not null))

DELETE FROM user\_score WHERE songid in (SELECT userId,COUNT(\*),songid FROM user\_score GROUP BY UserID HAVING COUNT(\*)=1)

DELETE FROM user\_score WHERE userid in (SELECT userid from(SELECT userid FROM user\_score GROUP BY UserID HAVING COUNT(\*)=1) as a)

insert into song(SongId,SongName,PlayTime,PubTime) values(451703096,'Shape of You',NULL,NUll) on duplicate key update SongName=VALUES(SongName),PlayTime=VALUES(PlayTime),PubTime=VALUES(PubTime)

SELECT userId,COUNT(\*) as total FROM user\_score GROUP BY UserID

SELECT uuser.userId,COUNT(\*) as total FROM user\_score,uuser WHERE uuser.UserID=user\_score.userId And flag="NoPermission" GROUP BY UserID ORDER BY total #DESC

SELECT \* from uuser where userarea is NULL and userid='100266803'

SELECT \* from uuser where userarea is NULL And flag="NoPermission"

insert into uuser(UserId,UserName,flag) values('100591478','静之丶','NoPermission')

SELECT \* FROM user\_score WHERE userid='249436803'

SELECT \* from uuser where userarea is NULL And flag!="NoPermission"

SELECT userID,MAX(songscore) FROM user\_score GROUP BY userid

SELECT songid FROM user\_score WHERE songscore=100

DELETE FROM user\_score WHERE userId IN (SELECT userid FROM(SELECT user\_score.userId from user\_score,uuser where uuser.userId=user\_score.userId AND uuser.flag='NoPermission') as a)

INSERT INTO user\_score(userID,songId,songscore)SELECT \* FROM sheet1 WHERE songid not IN(SELECT songid from user\_score) AND userid not IN(SELECT userid FROM sheet1 GROUP BY userid HAVING COUNT(\*)=1)

SELECT CommentTime,COUNT(\*) FROM comment GROUP BY commentTime

SELECT songId,count(\*) FROM song\_comment GROUP BY songId HAVING COUNT(\*) >500

INSERT INTO content Select commentID,commentcontent FROM comment WHERE commentid in(Select commentId FROM song\_comment WHERE songid='108242')

SELECT COUNT(\*) FROM song\_comment group by songid having songid='' and linkedcount between 1 and 100

SELECT count(\*) FROM `comment` WHERE commentid in(select commentid FROM song\_comment WHERE songid='28315784') and likedcount BETWEEN 1 and 100

SELECT \* from song,song\_score WHERE song.songid=song\_score.id;

insert into tagtag SELECT playlisthasstyle.StyleName FROM playlisthasstyle

SELECT Userid,UserAge FROM uuser WHERE UserID IN (SELECT distinct userid FROM user\_score WHERE songId in( SELECT songId from user\_score WHERE userid='100049561'))

SELECT userid FROM uuser WHERE flag!='NoPermission'

SELECT userid from uuser where userarea is NULL And flag!="NoPermission"

insert into uuser(UserId,UserName,UserAge,UserArea,flag) values('100003331','瑞恩杀西瓜',24.7735159817352,'北京市','Default') on duplicate key update UserArea=VALUES(UserArea),UserAge=VALUES(UserAge),UserName=VALUES(UserName),flag=VALUES(flag)

UPDATE uuser set UserAge='Unknown' and UserArea='Unknown' WHERE flag='NoPermission'

insert into uuser(UserId,UserName,UserAge,UserArea) values('100003331','',24.7735159817352,'北京市') on duplicate key update UserArea=VALUES(UserArea),UserAge=VALUES(UserAge)

DELETE FROM user\_score WHERE userid in(SELECT userId FROM uuser where flag='NoPermission')

SELECT \* from uuser where flag='NoPermission';

SELECT \* FROM uuser where userage is null

SELECT UserAge FROM uuser,user\_comment WHERE uuser.UserID=user\_comment.userId AND user\_comment.commentId IN (SELECT commentId FROM song\_comment WHERE songId='27743746')

SELECT UserArea FROM uuser,user\_comment WHERE uuser.UserID=user\_comment.userId AND user\_comment.commentId IN (SELECT commentId FROM song\_comment WHERE songId='27743746')

SELECT UserArea FROM uuser GROUP BY UserArea

UPDATE Uuser SET UserArea=REPLACE(UserArea,' ','')

UPDATE Uuser SET UserArea='Unknown' WHERE UserArea=''

SELECT stylename, count(\*) count from playlisthasstyle GROUP BY playlisthasstyle.StyleName

SELECT \* FROM user\_score WHERE songscore>55;

SELECT \* FROM song,tempus where song.songid=tempus.songid;

SELECT \* FROM song WHERE songid='36496637'

## 数据获取与清洗

### 设计思路

　　为提高数据爬取速度，我们采用了Scrapy开源框架来爬取数据。网易云音乐的用户界面有用户的听歌排行，根据推荐算法，我们只需要爬取用户听歌排行，根据用户对排行榜上每首歌曲的听歌频率，得出用户对于每首歌的评分，将用户id，歌曲id，评分爬取下来，把用户的其他信息和歌曲的信息一同爬取下来，方便后期的可视化统计。

### 代码展示

|  |
| --- |
| *# -\*- coding: utf-8 -\*-* **import** scrapy **import** a\_functions **from** scrapy.http **import** FormRequest **import** json **from** music.items **import** RateItem **import** time **import** sys **import** a\_functions **class** UserItemRateSpider(scrapy.Spider):  name = **'user\_item\_rate'** custom\_settings = {  **'ITEM\_PIPELINES'**:{**'music.pipelines.RatePipeline'**: 400},  }  allowed\_domains = [**'music.163.com'**]  start\_urls = [**'https://music.163.com/'**]  **def** \_\_init\_\_(self,lrange=**'0'**,gap=**'50'**, \*args, \*\*kwargs):  super(UserItemRateSpider,self).\_\_init\_\_(\*args, \*\*kwargs)  self.lrange = int(lrange)  self.rrange = self.lrange + int(gap)  self.start\_urls=[**'https://music.163.com/'**]   **def** read\_user(self):  conn = a\_functions.open\_db(**'localhost'**, **'root'**, **'2436399'**, **'music163'**)  users = a\_functions.query\_db(conn, **'SELECT DISTINCT userid from user\_score where songid in(SELECT songid FROM song where songid in(SELECT songid FROM user\_score GROUP BY songid ) and PlayTime is null)'**)  userids = []  **for** u **in** users:  userids.append(u[0])  print(len(userids))  *# if self.rrange>len(userids):  # self.rrange = len((userids))  # print('l:'+str(self.lrange)+',r:'+str(self.rrange))* **return** userids   **def** from\_timestamp\_to\_date(self, time\_stamp, format):  time\_stamp = time\_stamp \* 0.001  real\_date = time.strftime(format, time.localtime(time\_stamp))  **return** real\_date   **def** parse(self, response):  users = self.read\_user()*#17188 48703 44003 25015 25015* print(users)  **for** u **in** users:  item = RateItem()  item[**'user'**] = u  print(**'user '**+str(u)+**'start parse'**)  priKey = a\_functions.create\_random\_16()  data = **'{uid:"'**+u+**'",type:"-1",limit:"1000",offset:"0",total:"true",csrf\_token:""}'** params = a\_functions.get\_params(data,priKey)  encSecKey = a\_functions.get\_encSecKey(priKey)  data = {**"params"**: str(params, encoding=**'utf-8'**), **"encSecKey"**: encSecKey}  **yield** FormRequest(url=**'https://music.163.com/weapi/v1/play/record?csrf\_token='**,formdata=data, callback=self.next,method=**'POST'**,meta={**'it'**:item})  print(**'user '**+str(u)+**'end parse'**)  **def** next(self,response):  da = response.body.decode(**'utf-8'**)  data = json.loads(da)  item = response.meta[**'it'**]  **if 'msg' in** data:  db = a\_functions.open\_db(**'localhost'**, **'root'**, **'2436399'**, **'music163'**)  id = item[**'user'**]  print(str(id)+**" No Permission!"**)  sql = **'select username from uuser where userid='**+str(id)  data = a\_functions.query\_db(db,sql)  print(data[0][0])  a\_functions.exe\_db(db,**'insert into uuser(UserId,UserName,flag) values(%s,%s,%s) on DUPLICATE KEY UPDATE UserName=VALUES(UserName),flag=VALUES(flag)'**,(str(id),data[0][0],**'NoPermission'**))  db.close()  **else**:  allData = data[**'allData'**]  **for** da **in** allData:  item[**'score'**] = da[**'score'**]  song = da[**'song'**]  song = song[**'song'**]  print(**'song'**+str(song[**'id'**])+**'parse begin'**)  item[**'id'**] = song[**'id'**]  item[**'name'**] = song[**'name'**]  item[**'singers'**]=[]  **for** s **in** song[**'artists'**]:  singer = {}  singer[**'id'**] = s[**'id'**]  singer[**'name'**] = s[**'name'**]  item[**'singers'**].append(singer)  t = song[**'pubTime'**]  item[**'pubTime'**] = self.from\_timestamp\_to\_date(t,**'%Y-%m-%d'**)  item[**'playTime'**] = song[**'playTime'**]  **yield** item  print(**'song'**+str(song[**'id'**])+**'parse end'**)  **class** RatePipeline(object):  **def** \_\_init\_\_(self):  self.conn = a\_functions.open\_db(**'localhost'**, **'root'**, **'2436399'**, **'music163'**)   **def** process\_item(self, item, spider):  **try**:  song\_id = item[**'id'**]  print(**'song '**+str(song\_id)+**'start ratepipline'**)  print(item)  song\_name = item[**'name'**]  song\_score = item[**'score'**]  user\_Id = item[**'user'**]  playTime = item[**'playTime'**]  pubTime = item[**'pubTime'**]  artists = item[**'singers'**]  a\_functions.exe\_db(self.conn, **'insert into song(SongId,SongName,PlayTime,PubTime) values(%s,%s,%s,%s) on duplicate key update SongName=VALUES(SongName),PlayTime=VALUES(PlayTime),PubTime=VALUES(PubTime)'**, (song\_id, song\_name,playTime,pubTime))  **for** a **in** artists:  a\_functions.exe\_db(self.conn, **'insert into artist(ArtistId,ArtistName) values(%s,%s) on duplicate key update ArtistName=VALUES(ArtistName),'**,  (a[**'id'**],a[**'name'**]))  a\_functions.exe\_db(self.conn, **'REPLACE INTO SongHasArtist(SongId,ArtistId) VALUES (%s,%s)'**,  (song\_id, a[**'id'**]))  a\_functions.exe\_db(self.conn, **'replace into user\_score(UserId,SongId,SongScore) values(%s,%s,%s)'**, (user\_Id, song\_id, song\_score))  print(**'song '**+str(song\_id)+**'end ratepipline'**)  **except** Exception **as** e:  traceback.print\_exc()  **return** item   **def** close\_spider(self, spider):  self.conn.close() |

## 推荐算法应用

### 设计思路

　　我们采用的推荐算法是基于Python的开源推荐系统项目Surprise内的协同过滤算法，其基于最为基本的机器学习算法——K近邻算法。其模型的训练输入为用户id、歌曲id、用户听歌频率比，训练后即得出与目标用户或目标歌曲最为接近的指定数量的对象。通过用户与前台的互动，获取用户对推荐歌曲的评价，以达到改良推荐效果的目的。

　　推荐流程如下：

　　用户输入其id，若其id不在库中，则尝试当场爬取其听歌历史；否则直接从数据库中读取相关信息：

　　1. 若能获取id听歌历史，则显示其听歌榜单，并显示

　　(1) 每首喜欢该歌的用户还喜欢哪些歌

　　(2) 这些歌曲被收在哪些歌单里

　　(3) 根据协同过滤算法，推荐其可能喜欢的若干首歌，

　　(4) 与其喜好相近的6个用户并显示这些用户的听歌排行；

2. 若不能获取id历史，则推荐热门榜单里的若干首歌。

### 代码展示

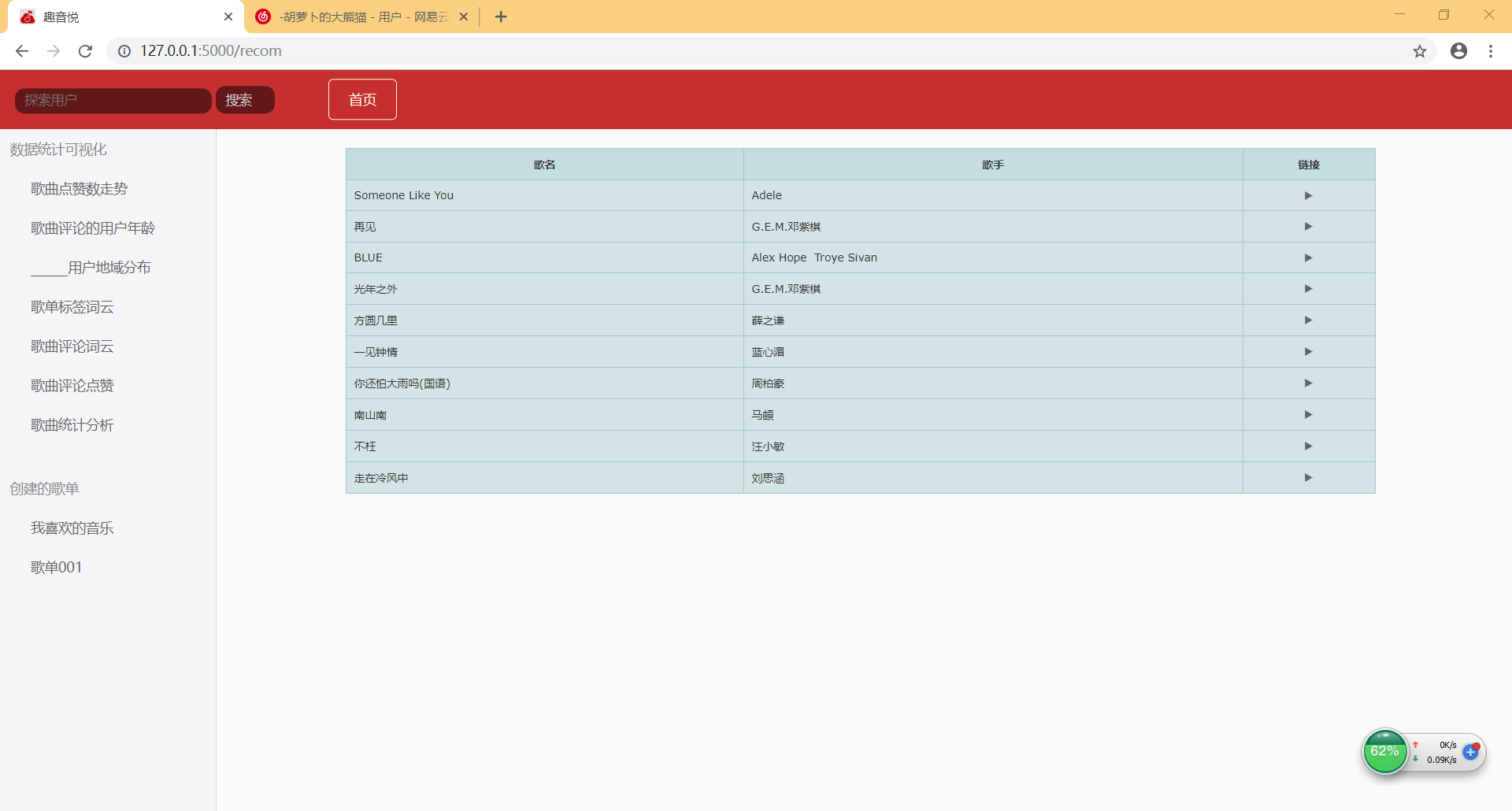
|  |
| --- |
| #获得歌名到id的映射  def read\_item\_name():  (db,cursor) = connectdb()  sql='SELECT songid,songname FROM song WHERE songid IN (SELECT songid FROM user\_score)'  cursor.execute(sql)  name\_id = cursor.fetchall()  sid\_to\_name = {}  sname\_to\_rid = {}  for item in name\_id:  sid\_to\_name[item['songid']]=item['songname']  sname\_to\_rid[item['songname']] = item['songid']  return sid\_to\_name,sname\_to\_rid  #训练模型  def get\_model(user\_based=False):  #载入用户歌曲评分数据  reader = Reader(sep=',',rating\_scale=(1,100),line\_format='user item rating')  data = Dataset.load\_from\_file(r'C:/users/89556/desktop/user\_score.txt',reader)  #建立训练集  train\_set = data.build\_full\_trainset()  #使用pearson\_baseline方式计算相似度 False以item为基准计算相似度 本例为电影之间的相似度  sim\_options = {'name':'pearson\_baseline','user\_based':user\_based}  #使用KNNBaseline算法  algo = KNNBaseline(sim\_options=sim\_options)  #训练模型  algo.train(train\_set)  print("Done model")  return algo  algos = get\_model()  # algou = get\_model(True)  id2name,name2id=read\_item\_name()  #基于模型做推荐  def showSimiliarSong(algo,id\_name=id2name,number=10,id='16435049'):  #转换成innerId  print('algo'+str(algo))  only\_one\_inner\_id = algo.trainset.to\_inner\_iid(id)  #通过模型获得推荐歌曲  only\_one\_ne = algo.get\_neighbors(only\_one\_inner\_id,number)  #转换成songId  ne\_raw\_id = [algo.trainset.to\_raw\_iid(innerid) for innerid in only\_one\_ne]  return ne\_raw\_id  @app.route('/recom', methods=['POST'])  def recom():  global algos  print('algos'+str(algos))  algon = algos  global id2name  global name2id  # global algou  global lastuser  data = request.form  user = lastuser  print('user'+user)  if data['user']:  user = data['user']  lastuser = user  sql = 'select \* from user\_score where userid=%s order by songscore desc'  (db,cursor) = connectdb()  cursor.execute(sql,(str(user)))  ids = cursor.fetchall()  dataStatics={}  dataStatics['data']=[]  flag=True  if not ids:  url='https://music.163.com/weapi/v1/play/record?csrf\_token='  priKey = a\_functions.create\_random\_16()  data = '{uid:"'+user+'",type:"-1",limit:"1000",offset:"0",total:"true",csrf\_token:""}'  params = a\_functions.get\_params(data,priKey)  encSecKey = a\_functions.get\_encSecKey(priKey)  data = {"params": str(params, encoding='utf-8'), "encSecKey": encSecKey}  postinfo = a\_functions.fake\_browserp(url,data)  info = json.loads(postinfo)  if 'msg' in data:  a\_functions.exe\_db(db,'insert into uuser(UserId,UserName,flag) values(%s,%s,%s) on DUPLICATE KEY UPDATE UserName=VALUES(UserName),flag=VALUES(flag)',(user,'Default','NoPermission'))  flag=False  else:  alldata = info['allData']  text = ''  for i in range(0,len(alldata)):  score = alldata[i]['score']  songinf = alldata[i]['song']['song']  songname = songinf['name']  songid = songinf['id']  if i == 0:  text = text + '\n'  id = songid  text = text + str(user)+','+str(songid)+','+str(score)  if i != len(alldata)-1:  text = text + '\n'  artists = []  for ar in songinf['artists']:  artist={}  artist['id']=ar['id']  artist['name']=ar['name']  artists.append(artist)  sql = 'insert into song(songid,SongName) values(%s,%s) on duplicate key update SongName=VALUES(SongName)'  cursor.execute(sql,(songid,songname))  a\_functions.exe\_db(db,'insert into uuser(UserId,UserName,flag) values(%s,%s,%s) on DUPLICATE KEY UPDATE UserName=VALUES(UserName),flag=VALUES(flag)',(user,'Default','NoPermission'))  for a in artists:  a\_functions.exe\_db(db, 'insert into artist(ArtistId,ArtistName) values(%s,%s) on duplicate key update ArtistName=VALUES(ArtistName),',  (a['id'],a['name']))  a\_functions.exe\_db(db, 'REPLACE INTO SongHasArtist(SongId,ArtistId) VALUES (%s,%s)',  (songid, a['id']))  a\_functions.exe\_db(db, 'replace into user\_score(UserId,SongId,SongScore) values(%s,%s,%s)', (user, songid, score))  with open(r'C:/Users/89556/Desktop/user\_score.txt','a') as info:  info.write(text)  info.close()  algon= get\_model()  else:  id=ids[0]['songid']  if flag:  print('id'+str(id))  print('algon:'+str(algon))  similarid=showSimiliarSong(algo=algon,number=10,id=id)  else:  dataStatics['data']=[]  re = a\_functions.query\_db(db,'SELECT song.songid,count(\*) count FROM song,user\_score where song.songid=user\_score.songid and PlayTime is not null GROUP BY song.songid ORDER BY count DESC limit 30')['']  similarid = []  for r in re:  similarid.append(r[0])  for id in similarid:  sql = 'select songname from song where songid=%s'  cursor.execute(sql,(id))  da = cursor.fetchall()[0]  sql2 = 'select artist.artistname from songhasartist,artist where songhasartist.artistid=artist.artistid and songid=%s'  cursor.execute(sql2,(id))  singers = cursor.fetchall()  song={}  song['id']=id  song['name']=da['songname']  song['artist']=[]  for singer in singers:  song['artist'].append(singer['artistname'])  dataStatics['data'].append(song)  return render\_template('recom.html', statics=dataStatics)  @app.route('/feed',methods=['POST'])  def feed():  global algos  data = request.data  data = data.decode('utf-8')  print(data)  datas = data.split('&')  if len(datas)>1:  user = datas[-1].split('=')[-1]  print("user:"+user)  ids=[]  for i in range(0,len(datas)-1):  ids.append(datas[i].split('=')[-1])  print("Data:"+str(ids))  (db,cursor) = connectdb()  for id in ids:  sql = 'select \* from user\_score where songid=%s and userid=%s'  cursor.execute(sql,(id,user))  res = cursor.fetchall()  print(res)  sql = 'insert into user\_score(userid,songid,songscore) values(%s,%s,%s) on DUPLICATE KEY UPDATE songscore=songscore+%s'  cursor.execute(sql,(user,id,50,5))  sql = 'SELECT songscore FROM user\_score where userid=%s and songid=%s'  cursor.execute(sql,(user,id))  score = cursor.fetchone()['songscore']  info = open(r'C:/users/89556/desktop/Music/statics/data/user\_score.txt','a')  info.write("\n"+str(user)+","+str(id)+","+str(score))  info.close()  algos= get\_model()  return jsonify({'msg':'success'}) |

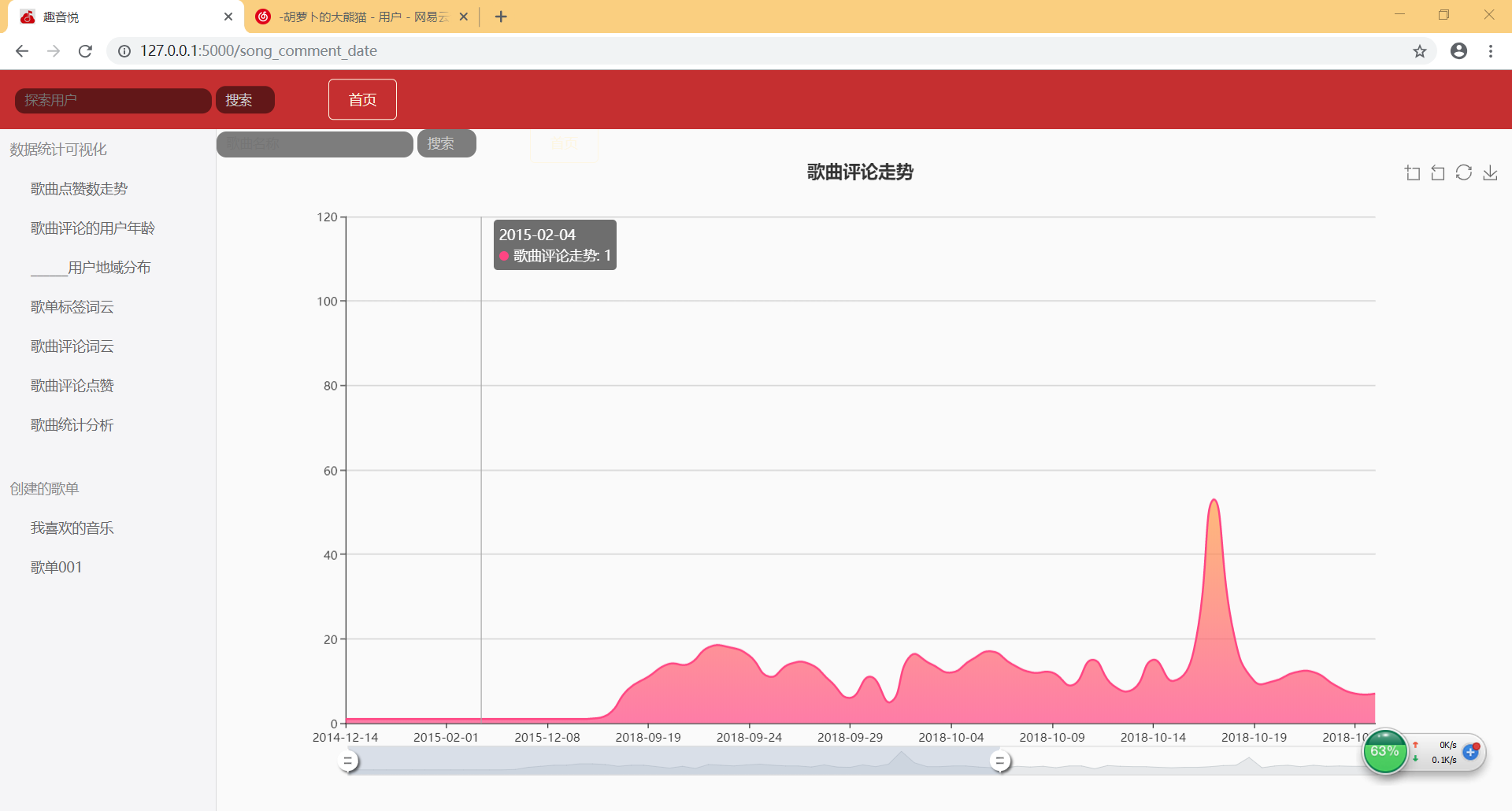
## 界面显示与数据可视化

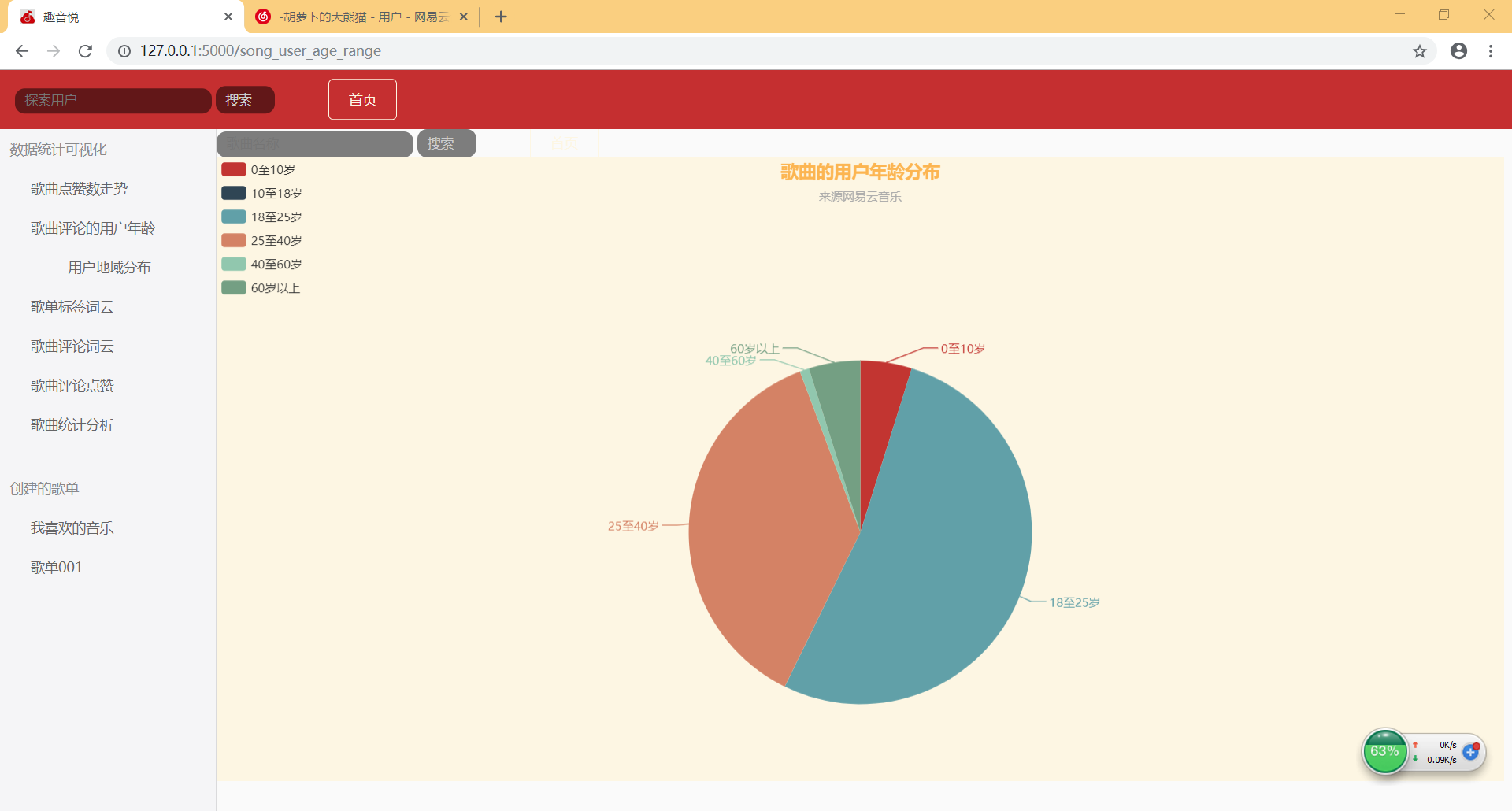
### 设计思路

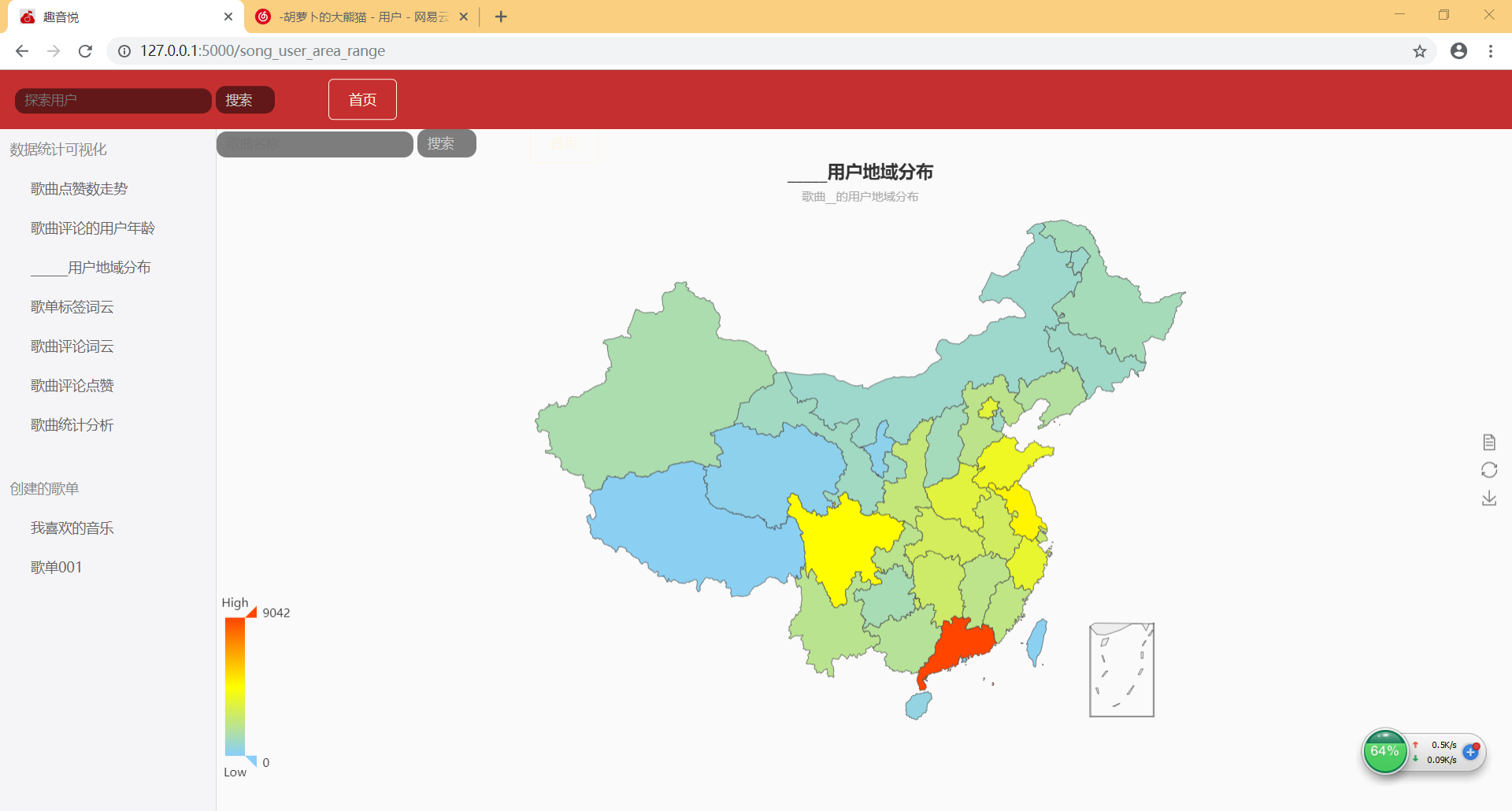
　　根据前期爬取的用户数据，歌曲评论数据，做相应的统计分析，使用Echarts做前端显示，Flask做后端与前端的数据传送。

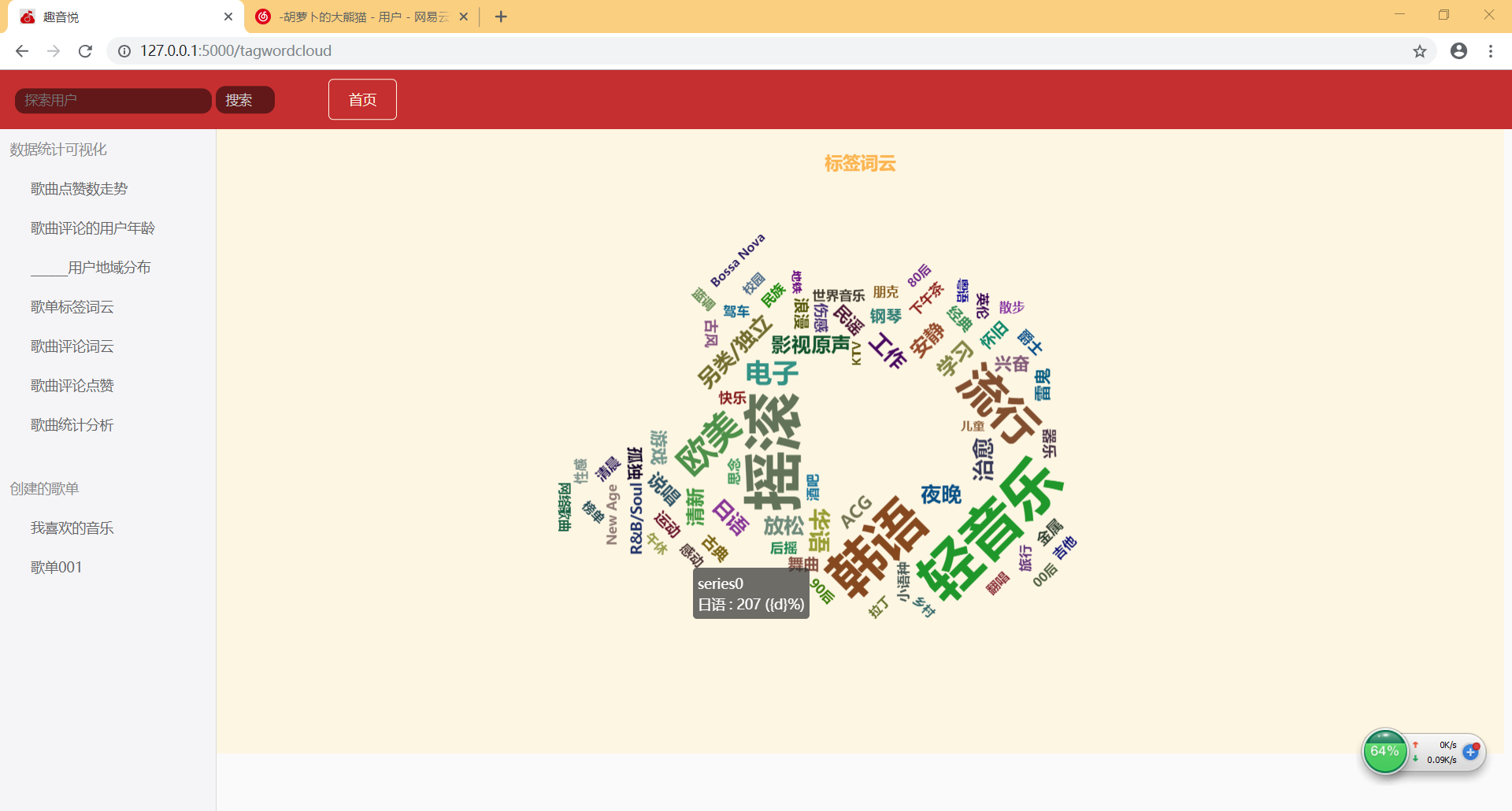
基本界面如下：

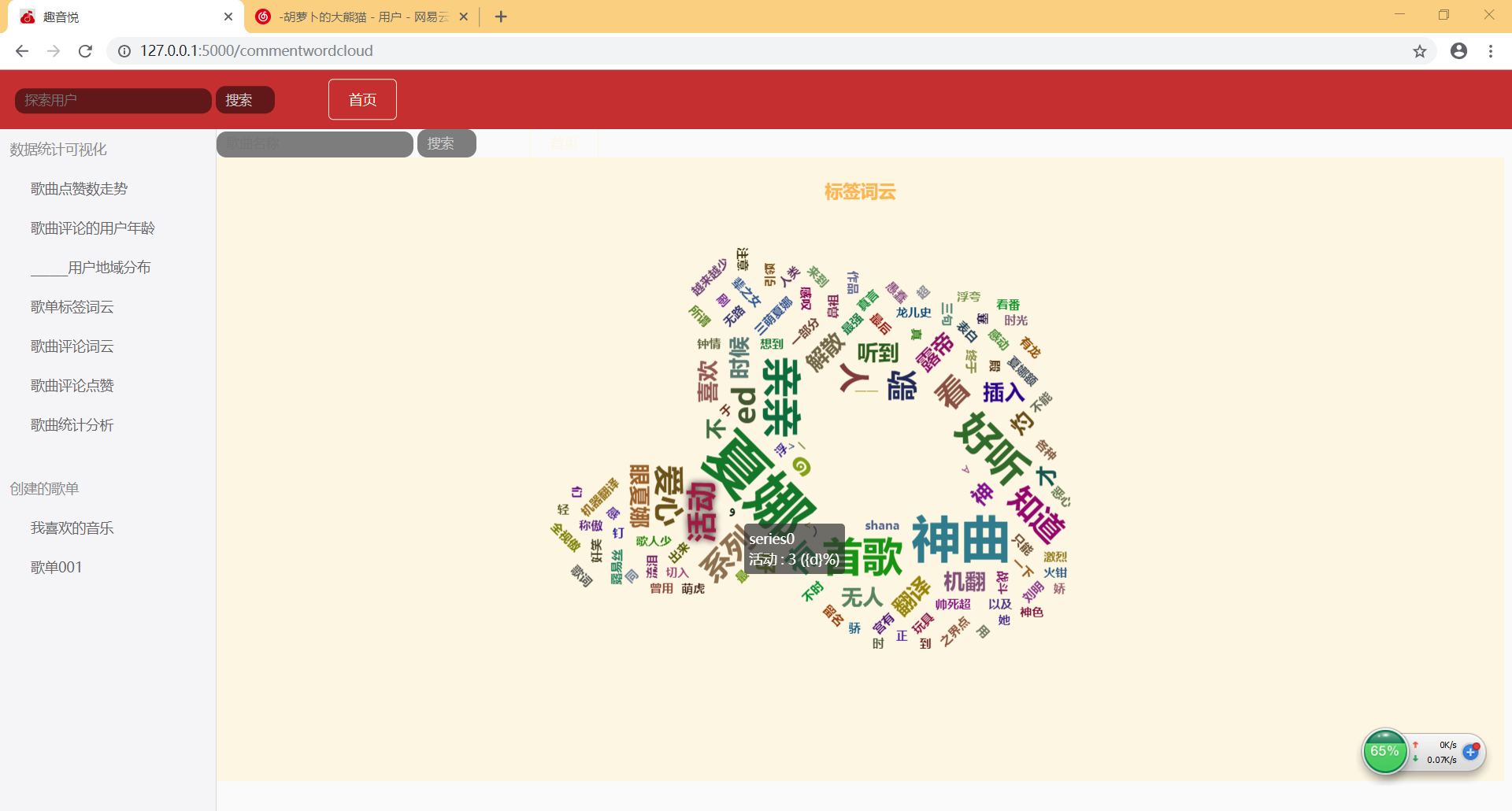


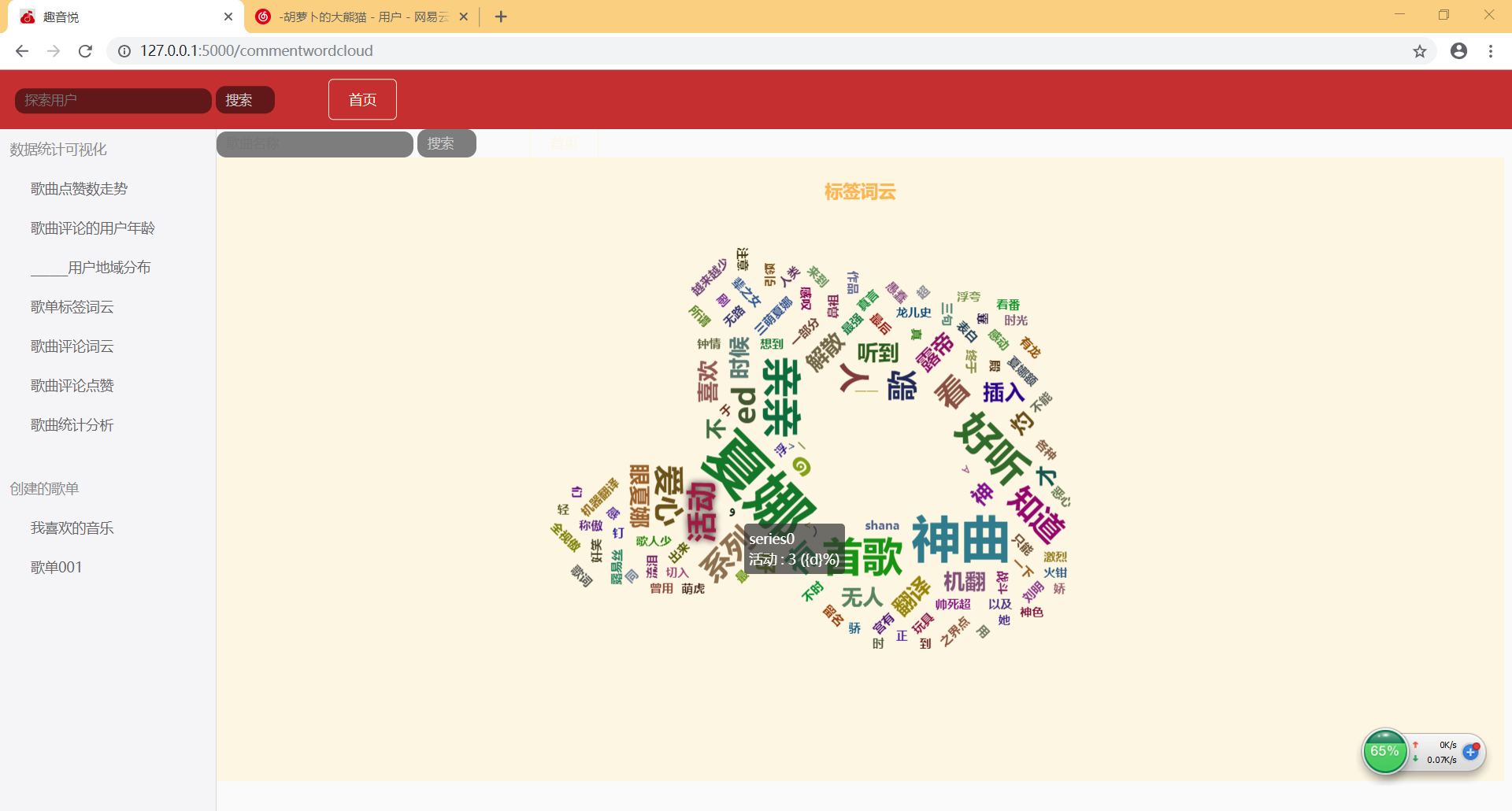


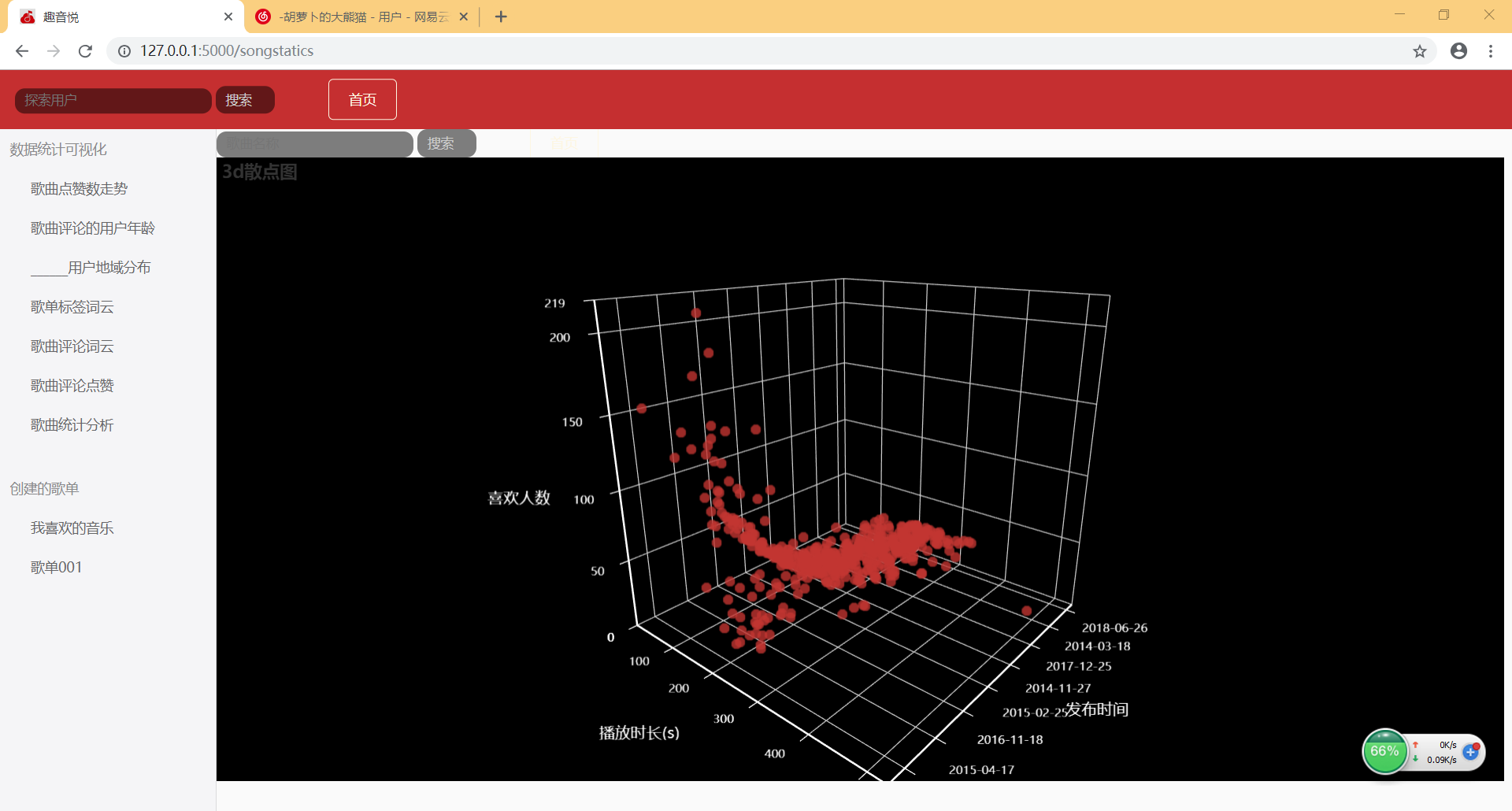












### 代码展示

|  |
| --- |
| {% extends 'layout.html' %}  {% block body %}  <div class="menu\_search">  <input type="text" placeholder="歌曲名称" autofocus="" id="inpSearch" class="search\_inp" />  <input type="button" value="搜索" id="seaSearch" class="search\_sea" />  <a id='topbar'href="{{url\_for('indexb')}}">首页</a>  </div>  <div id="item" class="main\_container">  </div>  <script type="text/javascript">  $(document).ready(function() {  var myChart = echarts.init(document.getElementById('item'));  data = {{ statics | tojson }};  option = {  backgroundColor: '#FDF6E3',  title: {  text: '歌曲点赞分布',  subtext: '来源网易云音乐',  x: 'center',  textStyle: {  color: '#FCB551'  }  },  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  legend: {  orient: 'vertical',  left: 'left',  data: ['0~10赞', '10~50赞', '50~100赞', '100~500赞', '500~1000赞', '1000赞以上']  },  series: [{  name: '访问来源',  type: 'pie',  radius: '55%',  center: ['50%', '60%'],  data: [  { value: data['0~10'], name: '0~10赞' },  { value: data['10~17'], name: '10~50赞' },  { value: data['18~25'], name: '50~100赞' },  { value: data['26~40'], name: '100~500赞' },  { value: data['40~59'], name: '500~1000赞' },  { value: data['60~'], name: '1000赞以上' }  ],  itemStyle: {  emphasis: {  shadowBlur: 10,  shadowOffsetX: 0,  shadowColor: 'rgba(0, 0, 0, 0.5)'  }  }  }]  };  myChart.setOption(option);  })  // 基于准备好的dom，初始化echarts实例  </script>  {% endblock %}  {% extends 'layout.html' %}  {% block body %}  <div class="menu\_search">  <input type="text" placeholder="歌曲名称" autofocus="" id="inpSearch" class="search\_inp" />  <input type="button" value="搜索" id="seaSearch" class="search\_sea" />  <a id='topbar'href="{{url\_for('indexb')}}">首页</a>  </div>  <div id="item" class="main\_container">  </div>  <!-- scripts -->  <script>  var img = new Image();  img.src='../static/images/0.jpg'  var chart = echarts.init(document.getElementById('item'));  data = {{ statics | tojson }}  var option = {  backgroundColor: '#FDF6E3',  title: {  text: '标签词云',  left: 'center',  top: 20,  textStyle: {  color: '#FCB551'  }  },  tooltip: {  trigger: 'item',  formatter: "{a} <br/>{b} : {c} ({d}%)"  },  series: [{  type: 'wordCloud',  // The shape of the "cloud" to draw. Can be any polar equation represented as a  // callback function, or a keyword present. Available presents are circle (default),  // cardioid (apple or heart shape curve, the most known polar equation), diamond (  // alias of square), triangle-forward, triangle, (alias of triangle-upright, pentagon, and star.  shape: 'circle',  // A silhouette image which the white area will be excluded from drawing texts.  // The shape option will continue to apply as the shape of the cloud to grow.  maskImage: img,  // Folllowing left/top/width/height/right/bottom are used for positioning the word cloud  // Default to be put in the center and has 75% x 80% size.  left: 'center',  top: 'center',  width: '70%',  height: '80%',  right: null,  bottom: null,  // Text size range which the value in data will be mapped to.  // Default to have minimum 12px and maximum 60px size.  sizeRange: [12, 60],  // Text rotation range and step in degree. Text will be rotated randomly in range [-90, 90] by rotationStep 45  rotationRange: [-90, 90],  rotationStep: 45,  // size of the grid in pixels for marking the availability of the canvas  // the larger the grid size, the bigger the gap between words.  gridSize: 8,  // set to true to allow word being draw partly outside of the canvas.  // Allow word bigger than the size of the canvas to be drawn  drawOutOfBound: false,  // Global text style  textStyle: {  normal: {  fontFamily: 'sans-serif',  fontWeight: 'bold',  // Color can be a callback function or a color string  color: function() {  return 'rgb(' + [  Math.round(Math.random() \* 160),  Math.round(Math.random() \* 160),  Math.round(Math.random() \* 160)  ].join(',') + ')';  }  },  emphasis: {  shadowBlur: 10,  shadowColor: '#333'  }  },  // Data is an array. Each array item must have name and value property.  data: data['data']  }]  };  //chart.setOption(option);  img.onload = function() {  chart.setOption(option); //生成云图  };  </script>  {% endblock %} |

# 项目总结

## 岳名扬

在此次《数据分析与处理软件开发综合实践》中，小组在经过商议后，选择了网易云音乐作为数据来源。我主要和何士豪负责进行爬虫编写和推荐系统应用部分的工作。

结合我们前期的需求分析，我们各自做了不同的爬虫尝试，我们首先尝试使用urllib作为爬虫框架，但发现其不能处理网页中的ajax请求，而网易云音乐中许多有效的信息均存在于此。这时，我们发现了网页自动化测试框架selenium，其可以模拟浏览器向网页发送请求，并在网页加载的所有异步请求完成后获取网页内容，经尝试其确实可以获得内容，但每次驱动浏览器将花费大量时间，效率过低。最后，我们对网页异步请求API进行了分析后，使用了爬虫框架Scrapy，其支持多线程爬虫，并同时提供相应方法实现所获数据的存储，我们基于此实现了数据的初步获取。

在获取到一定数据后，我们进行了若干清洗工作，包括去除省级行政区全名中的“省”、“市”字样、去除评论中的标点符号等，以便在后期推荐以及数据可视化时使用这些数据。

在推荐系统方面，我们使用了Python开源推荐系统框架Surprise，应用了其所实现的协同过滤算法，通过读入指定格式的文件，其方法自动实现相似度矩阵的构建，我们也基于此实现了基于物品的协同过滤方法的歌曲推荐。而对于某些不能获取到听歌历史的用户，我们则推荐其近期热门歌曲。

本次综合实践让我更多地接触到了Python语言，发现了其丰富的支持库为我们的程序实现带来了极大的便利，也了解了许多数据分析与处理方面的知识。本次综合实践的成品虽不完美，比如存在程序存在大量代码冗余、数据库内待分析数据量较少等问题；但个人认为，已在很大程度上达到了我们的心理预期。而在开发过程中，与我同分工的何士豪同学在功能的实现上提出了许多新颖、可行的想法，让我深受启发；而他对这个项目的热忱以及认真实干的态度也深深地感染了我。这让我意识到，固然技术是开发人员的立身之本，但良好的合作和沟通、对工作的热爱和深入透彻的思考也是一个项目达标完成的必要条件。感谢本次综合实践的指导老师以及我的其他组员们在这个项目上对我的无私帮助和关怀，希望我以后能锻炼出更强的个人能力，并且心无旁骛地享受于项目开发的过程，在团体内发挥自己更多的能量。

## 袁名扬

在此次项目中我主要负责数据可视化中的部分内容，虽说这一次的重点是在爬虫爬取和分析数据，但是想要较为出色的完成一个项目，每一个模块都是必不可少的元素。所以本组成员都对自己负责的模块有着足够的重视。直至今日此次项目已经基本完工，我感觉收获颇多。由于此次项目主要在前端使用javascript，后台使用python来搭建，对于小组各成员和我本人来说都是较为陌生的领域，所以在项目起初的一两个礼拜我们都是在网上通过相关网课和学习资料了解两种语言的语法和特性，并布置相关的项目环境。在实现数据可视化方面有很多插件和IDE能帮助我们实现具体功能，最终小组成员一致确定使用百度提供的开源js库来进行数据的可视化。

在查阅相关文档和使用教程以后，echarts的上手还是相对容易的，但难点在于如何连接数据库并把数据库中的数据通过sql语句成功提取出自己需要的部分并设置为echarts中对象的具体数据属性。开始我们是分头实现各自的功能，所以关于数据库连接方面也是各自想各自的办法。由于用node库的require来建立数据库连接相对简单，故我起初的想法就是通过调用node中的相关功能来实现连接，然而查阅相关资料以后发现node是用来编写专属程序的，若想在js中使用它的require还需要安装Browserrify并进行一系列的配置，由于时间紧迫，且本人对windows命令行的各种操作和webstormIDE的配置不够熟练，最终没能成功实现自己的想法，在数据库连接这部分最终由小组其他成员在python实现了连接和传递数据的接口。

这次任务由于个人技术的不成熟，基本功不够扎实，所以协助完成的工作不多。虽然这一次自己没有负责爬虫部分的代码编写，但是自己也利用了项目期间的时间自学了python的爬虫入门，并成功实现了爬取部分网站上的特定信息及如何保存下来。所以此次项目还是收获颇多的。学到了项目实践的新知识，也对良好的团队协作对于整个项目的重要性有了更深刻的体会。没有小组成员之间频繁的沟通交流，互帮互助，积极协作，就不可能有更高的效率和更好的完成度。以后我在打磨个人编程能力，增强知识储备的同时，也会更加注重团队合作，争取成为一名更好的小组成员。

## 杜玉楷

本次课设主要分成了爬虫爬数据，数据处理，可视化三部分。我负责的是可视化中的部分统计功能。首先在概要设计阶段，我们在学习一些必须的技能，为详细设计做准备和铺垫工作。学习的是全栈工程师的养成，包括了爬虫，数据清洗处理，R语言进行可视化。

最一开始，我们想用R语言进行可视化，也一直在学习在R语言环境下将数据转换成图形，也基本能绘制出来想要的效果。但是后面遇到一个问题，就是涉及到用什么语言获取数据的问题，因为后台采用python爬取数据并存入数据库中，因此如果用r语言就还要与web,sql结合。显然学习成本高昂，系统稳定性不能保证。于是参照其他例子，选取使用echarts进行可视化，这样无论使用java或者python，都可以进行可视化。因为echarts其实就是百度前端团队的一个开源项目，本质为JavaScript，专门用以数据可视化。

详细设计阶段，主要就是设计一些sql语句，将需要统计的数据得到。还有一些数据库接口这些东西，因为当时还未确定采用什么语言开发，我当时采用的java环境下连接数据库查询数据，当然使用python其实就是一些语法需要更改，这些其实就是一些小问题了。然后就是把数据放进echarts里。因为需要绘制的图形的特性，当时就是选择使用饼状图形。一开始学习的过程中出现了一个大问题，就是直接写html文件，直接写进去示例代码，跑出来的效果没问题，但是在eclipseEE中运行就是一片空白，后面经查询发现就是还需要引入Jquery包，但是引入后在eclipseEE自带的浏览器上运行成功，在IE上运行又是一片空白，最后偶然因为组员采用的echarts，选择的不是下载引入echarts包，而是通过网络加载引入。这样就没有问题了。但是由于时间问题还是没有找到第一种方法的问题在哪里，这是一个比较可惜的地方。另外就是echarts真的很好用，在以后的项目中值得一试。

另外就是爬虫这一块我个人还是很感兴趣的，当时概要设计阶段也学了一些，但是因为这块比较难，个人能力有限，我负责的只是可视化，就没有学的很好。但是一些概念方这个同时对我以后的毕设有了初步方向的影响。最后就是感谢老师和我的小组，每次的课设都能让我收获一些新东西。

## 李琦

个人总结：这次课设我主要是负责数据库部分。通过这次程序设计让我对数据挖掘和分析有了一些初步的认识，美中不足的是自己因为最近比较繁忙没有参与爬虫与数据分析这部分的代码编写。这次程序设计让我对自己编程能力有了进一步的认识，使得之前所学知识在这次程序设计中得以练习，强化了自己各方面的能力，同时也了解了自己的不足。

这次课设中，作为组长也锻炼了我的组织能力，交流沟通能力，使得自己在组织成员方面更加得心应手，能够根好的团结组员分配任务，此外这次课设中岳铭扬与何士豪在这次程序设计中起了很大作用，因为在忙考研，所有由很大一部分工作由他俩代替我组织的，非常感谢他们，之后有机会的话我会更加努力当好组长。

## 黎政君

　　个人总结：通过这次的系统设计，我在很多方面都有所提高。综合运用所学的理论知识进行实际训练从而培养和提高了我独立工作的能力，巩固所学的知识，掌握系统程序的编排和运行，使自己的独立思考能力有了显著提高。本次课程设计使我更加了解一个程序系统的设计过程，也加强自己的程序设计能力。同时，通过网络资料的学习，了解了如何使用flask框架来开发网页以及如何使用echarts插件来可视化数据。

　　本次课设虽然完成，但却不完美，还有很多地方应该改进，比如界面的美化做得不好，UI是用户体验很重要的一部分，但自己没有重视这一点，只是在意功能部分，弱化了界面的美化。而且功能部分也还有需要增加和改进的部分。所有我打算在之后的时间里，系统地去学习html、css和JavaScript相关的知识。

## 何士豪

　　与我而言数据挖掘是一个非常有意思的过程，试想一下，从一群数据中分析出信息获取知识，利用数据的价值获取有价值的信息，创造更大的价值，这本是是一件特别有意义的事情，整个项目过程于我而言是非常享受的一个过程，遗憾的是，由于时间关系，没能把项目做得更完善，如果有时间，我会继续做下去。

## 鲁芯丝雨

　　在此次项目中，我主要参与了项目数据可视化部分，尽可能展现一个完整的项目尽自己的一份力。第一，由于这次的项目中有很多大家在以前没有学习过的领域，因此在组员的推荐下，我在实践前期观看了网上关于CSS、JS、Flask框架网页搭建、Echarts等相关教程课程，为接下来的数据可视化做准备。

在一开始搭建网页时由于之前不够熟悉Python和flask，我从安装python到安装各种插件花了一点时间，还出现了个别安装不成功的情况，这也让我烦恼良久，最后通过网上的解决方案，我直接下载了whl文件，修改了名字后直接安装才完成。

在完成安装后我试着运行.py文件结果一直不成功，我以为是自己安装过程不正确，最后在组员的建议下，我有安装了pycharm，可以较为直接地看见自己在哪个环节出了差错。同时，在编写网页中，也发现了python与java语言在HTML编写的些微不同，比如连接网页时，python是使用{{url\_for()}}函数，并且要在.py文件编写相关的函数。

在搭建网页时，css样式是一个特别繁琐的设计过程，因此我借鉴了网上的开源的CSS样式用在了自己的网页中，但是因为直接拿来用的话效果并不会直接出现，最后我在一一对应后，拿了自己看懂的CSS进行使用。

在视频课程中，我发现echarts的上手还是相对容易的,但难点在于如何连接数据库并把数据库中的数据通过sql语句成功提取出自己需要的部分并设置为echarts中对象的具体数据属性.这一点，我一直没有解决，最后还是交给了组里的同学。

在这个过程里，我再一次意识到了自己的不足，并且，在该系统的设计与分析过程中有了很多切身的体会和认识，更重要的是在学习和实践的过程里我还丰富了经验、增长了知识。下面主要是我在本次课程设计中的几点体会和看法：

这一次学习了之前没有接触过的语言、框架等，感觉到了自己能力的不足。在设计过程的时候，我们一边设计，一边讨论统一风格，使得努力探讨的问题得到解决。而设计，不会一步到位，需要不断补充和完善，而且，仍然存在很多问题在系统中未被发现，今后在实践应用中发现问题一定认真解决。

## 杨畅

　　工作总结：这次课设中，我和组长李琦主要负责的是数据库部分，数据库主要就是存储从网易云爬取的歌单，歌曲数据。并根据从网易云音乐获取的信息以及推荐算法和可视化需要，设计数据库关系。六个实体和实体间的关系，我们需要明确各个实体的功能和实体的关系。首先我们是一起商量绘制了E-R图确定了关系，所以相对于来说已经明确了各项关系后，建表时思路就会比较清晰不会混乱。主要实体是:艺人，歌曲，歌单，用户和评论。主要关系:艺人表演歌曲，歌曲属于下载歌单，歌曲包含评论，用户喜欢歌曲，用户发表评论，用户创建歌单。用户主要信息包含:用户ID,用户名，用户生日，用户所在地。艺人信息:艺人ID，艺人名。歌曲信息:歌曲ID，歌曲名，播放时长，发布时长等一系列表。

心得体会：

1.开始之前一定要首先明确具体需求，不然因为个人的理解差异可能会造成一定的偏差和错误所以要及时沟通及时止损，不然可能既没有完成需求还要浪费时间，我们的工作任务相对要在获取数据之后才能进行，没有遇到特别大的问题。任务比较常规，我只负责建了相关表格。

2.此次任务的数据库是在获得爬取数据基础上建立完善的。这样就需要很多的思考,讨论尽量可以完善各个实体的信息。从中我不仅巩固了以往的知识,学到了项目实践的新知识,也对良好的团队协作对于整个项目的重要性有了更深刻的体会。没有小组成员的数据提供就无法进行数据分析，无法建立数据库存储相关信息。同时也需要数据库存储才能在分析中提供数据。小组内沟通合作交流就必不可少了。通过课设，在增强知识储备的同时,也会更加注重团队合作。

# 组员成绩表

|  |  |
| --- | --- |
| **成员** | 成绩 |
| **岳名扬** | 优秀 |
| **袁名扬** | 优秀 |
| **杜玉楷** | 良好 |
| **李琦** | 其他 |
| **黎政君** | 其他 |
| **何士豪** | 良好 |
| **鲁芯丝雨** | 良好 |
| **杨畅** | 其他 |

# 参考文献

[1] 网易云音乐[EB/OL]. [2018-11-29]. https://music.163.com.

[2] Li Dengqing, Mei Honghui, Shen Yi, *et al*. ECharts: A Declarative Framework for Rapid Construction of Web-based Visualization[J]. Visual Informatics, 2018, 2(2), 136-146.

[3] NicolasHug, *et al*. Surprise[CP/OL]. [2018-11-29]. https://github.com/NicolasHug/Surprise.