对网易云音乐的数据挖掘与分析

杜涵悦
10181900144
华东师范大学

目录

1 引言

1.1 项目背景

1.1.1 选题背景

2019 年，恰逢新中国建国成立 70 周年，举国欢庆，热闹非凡。尤其在国庆期间，大街小巷都播放着《我和我的祖国》，气氛欢腾，一派融融。故突发奇想，想看看《我和我的祖国》这首歌曲在国庆期间的热度，并比较它与其他歌曲的热度区别。其次，该项目内容也符合个人的兴趣爱好，故抱有极大的热情进行该项目。

1.1.2 研究背景

近年来，网易云音乐的兴起和发展，引入了大量优质专辑和音乐，同时也吸引了大批用户，有了许多音乐和用户行为数据。该项目立足于挖掘网易云音乐的华语模块信息，对相关音乐和用户数据做可视化及分析。

1.2 研究方法与研究难点

1.2.1 方法

① 在 Windows 环境下，利用 pycharm（Python3.7）和 MySQL 来挖掘、存储数据。

② 调用 pyecharts、matplotlib 库来可视化数据。

1.2.2 难点

① 在数据挖掘方面：

网易云进行过更新，早先网上有的他人上传的爬取网易云音乐评论代码的 API 已不适用，因此，这一块的代码需自己修改重写，以绕过它的 AES 加密。

② 在数据可视化方面：

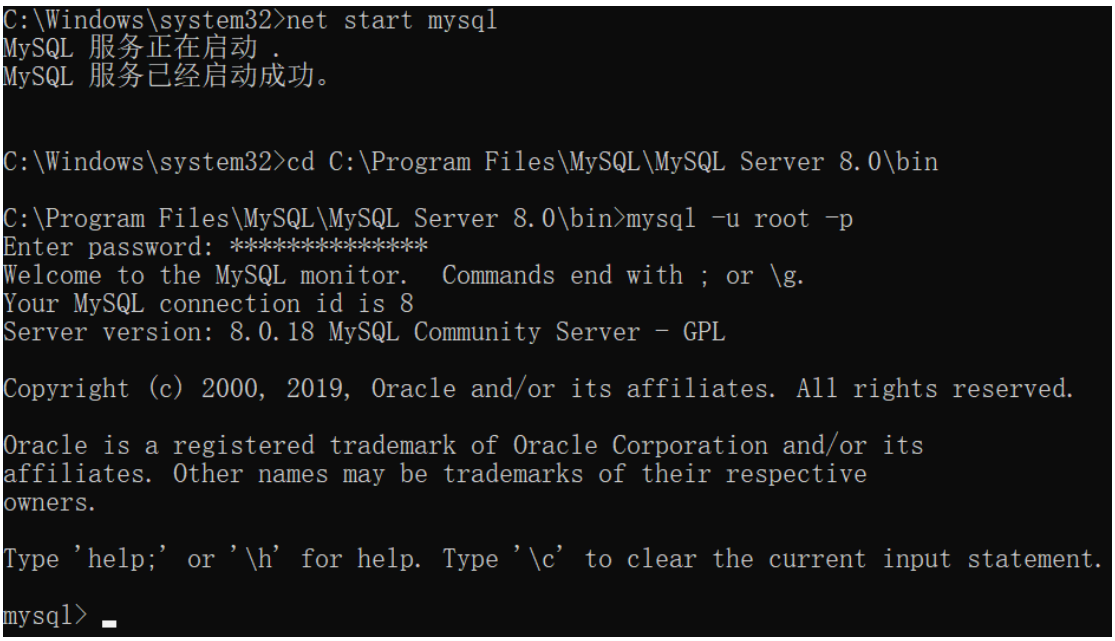
处理数据时，需对时间戳进行处理，还得转化地区编号与地区的对应关系。

2 数据采集与处理

2.1 数据库的构建

1、以管理员身份运行命令行，输入“net start mysql” 启动 MySQL。

之后切换至目录：C:\Program Files\MySQL\MySQL Server 8.0\bin ，“mysql -u root -p”输入密码后登录进入 root 用户的初始界面。



2、之后，进入 MySQL Workbench 8.0 CE，新建数据库名为 musicdb，在该数据库中新建 4 个 Table: artists，albums，musics 和 comments，分别存储歌手 ID（ARTIST_ID）、歌手名字（ARITST_NAME），歌手 ID、专辑 ID（ALBUM_ID），专辑 ID、歌曲 ID（MUSIC_ID）、歌曲名（MUSIC_NAME），歌曲 ID，该歌曲评论数（COMMENTS）。

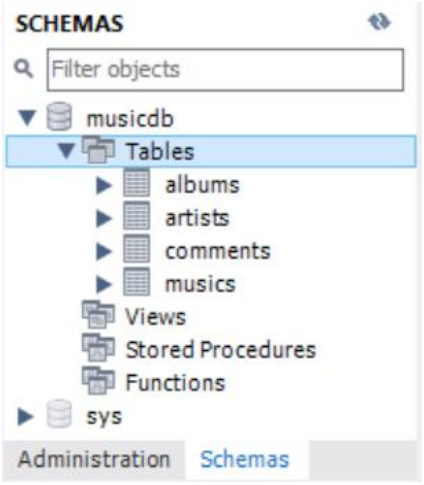


Table: artists

Columns:

ARTIST_ID	int(11)
ARTIST_NAME	varchar(90)

Table: albums

Columns:

ALBUM_ID	int(11)
ARTIST_ID	varchar(90)

Table: musics

Columns:

MUSIC_ID	int(11)
MUSIC_NAME	varchar(150)
ALBUM_ID	varchar(90)

Table: comments

Columns:

MUSIC_ID	int(11)
COMMENTS	int(11)

3、使用 gbk 编码，使新建好的 Table 中的 Columns 可以存储中文。

alter table artists modify ARTIST_NAME char(90) character set gbk;

同理，对 MUSIC_NAME 也进行该操作。

2.2 数据集的挖掘

此次实验的爬取过程中我们需将浏览器的 user-agent 信息复制到爬虫的 header 参数中，从而模拟浏览器。

2.2.1 爬取所有华语歌手的 ID

此步数据量较小，故采用单线程爬取，并将数据存入 MySQL 中。

```
gg = 1003

save_artist(gg, 0)
for i in range(65, 91):     # 26个字母
    save_artist(gg, i)
```

通过更改“gg”的值可设定爬取华语男歌手（gg=1001）、华语女歌手（gg=1002）和华语组合/乐队（gg=1003）的相关信息。

如下为获取和保存歌手 ID 的代码：

```
# 获取所有歌手的 ID
def get_all_artist():
    with connection.cursor() as cursor:
        sql = "SELECT `ARTIST_ID` FROM `artists` ORDER BY ARTIST_ID"
        cursor.execute(sql, ())
        return cursor.fetchall()

# 保存歌手
def insert_artist(artist_id, artist_name):
    with connection.cursor() as cursor:
        sql = "INSERT INTO `artists` (`ARTIST_ID`, `ARTIST_NAME`) VALUES (%s, %s)"
        cursor.execute(sql, (artist_id, artist_name))
    connection.commit()
```

2.2.2 根据歌手 ID 爬取专辑信息

此步数据量较大，但仍采用的是单线程，具体步骤与上一步爬取歌手 ID 时相似。

2.2.3 根据专辑 ID 爬取音乐信息

鉴于上一步采用单线程时爬取数据速度较慢，此步开始采用多线程。

这步需服从爬虫的礼貌性，尽量减少对服务器的负担，这也是为了不被反爬机制影响。所以选取合适的线程并让每个线程在爬取后 sleep 一段时间以防止因爬取速度过快而被反爬。

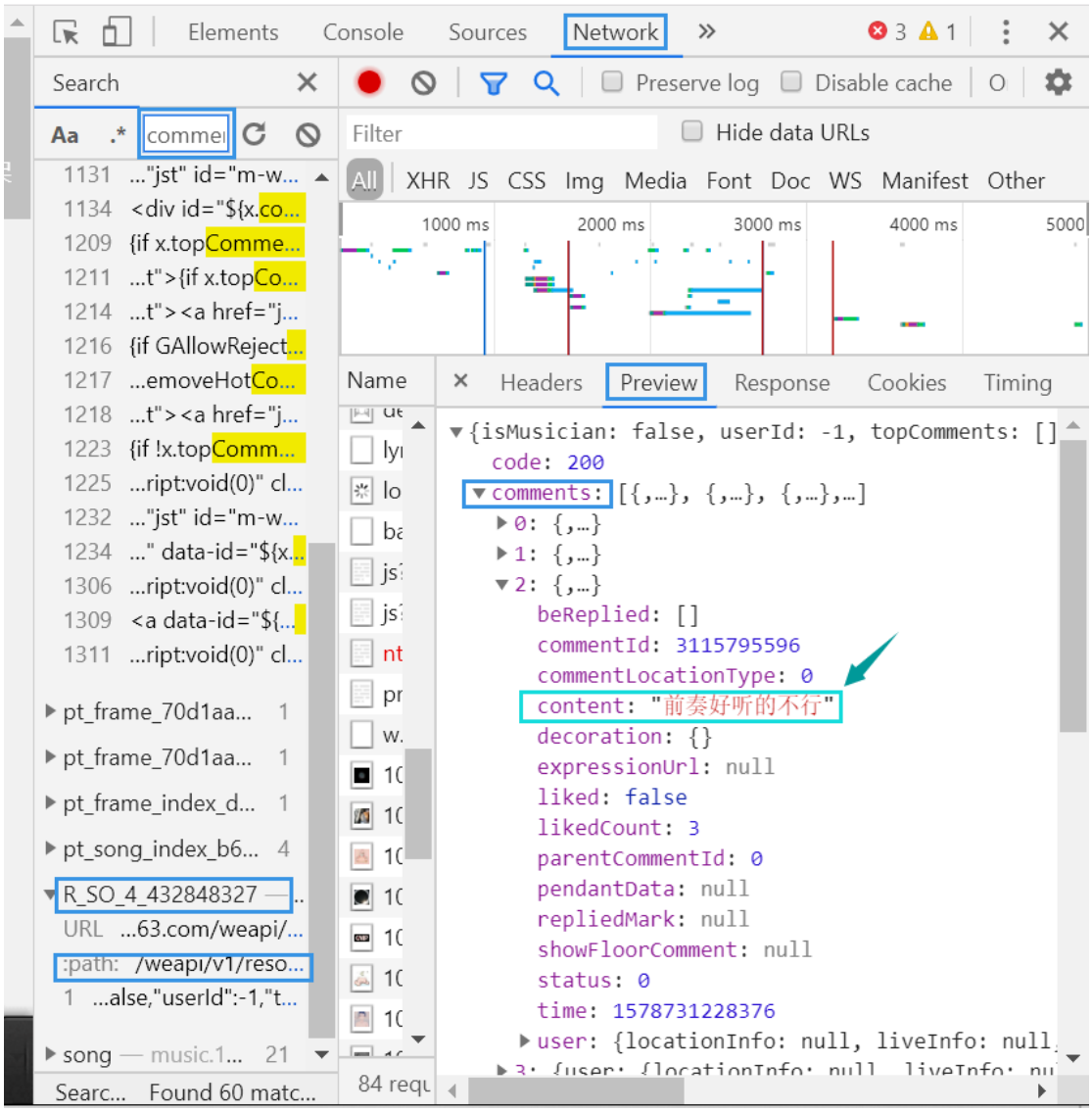
如下为该步的代码（从 Typora 导出）：

```
"""
根据专辑 ID 获取到所有的音乐 ID
"""
import requests
from bs4 import BeautifulSoup
import time
import sql
from queue import Queue
import threading

gLock = threading.Lock()

class Consumer(threading.Thread):
    def __init__(self, myQueue):
        super(Consumer, self).__init__()
        self.myQueue = myQueue
        self.headers = {
            'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8',
            'Accept-Encoding': 'gzip, deflate, sdch',
            'Accept-Language': 'zh-CN,zh;q=0.8,en;q=0.6',
            'Cache-Control': 'no-cache',
            'Connection': 'keep-alive',
            'DNT': '1',
            'Host': 'music.163.com',
            'Pragma': 'no-cache',
            'Referer': 'http://music.163.com/',
            'Upgrade-Insecure-Requests': '1',
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.143 Safari/537.36'
        }

    def saveMusic(self):
        self.album_id = self.myQueue.get(block=True)['ALBUM_ID']
        params = {'id': self.album_id}
        # 获取专辑对应的页面
        r = requests.get('http://music.163.com/album', headers=self.headers, params=params)
        # 网页解析
        soup = BeautifulSoup(r.content.decode(), 'html.parser')
        body = soup.body
        musics = body.find('ul', attrs={'class': 'f-hide'}).find_all
```

```
('li')  # 获取专辑的所有音乐
        for music in musics:
            music = music.find('a')
            music_id = music['href'].replace('/song?id=', '')
            music_name = music.getText()
            gLock.acquire()
            try:
                sql.insert_music(music_id, music_name, self.album_id)
            except Exception as e:
                # print(str(i) + ': ' + str(e))
                time.sleep(0.05)
            gLock.release()
    def run(self):
        while not self.myQueue.empty():
            self.saveMusic()


if __name__ == '__main__':
    albums = sql.get_all_album()
    my_music = Music()
    myQueue = Queue()
    for i in albums:
        myQueue.put(i)
    for i in range(500):
        t = Consumer(myQueue)
        t.start()
```

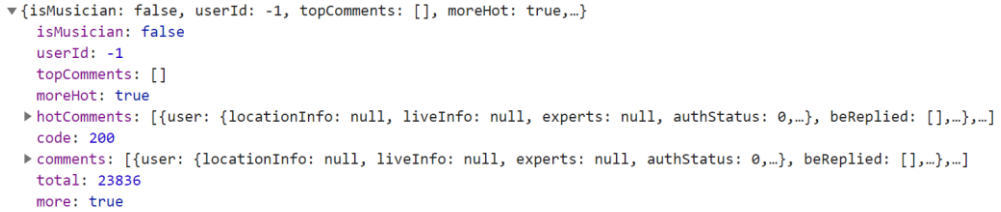
2.2.4 根据音乐爬取评论数

此步仍采用多线程。只需查找网页中带 comment 词语的资源即可找到所需的信息，即图示信息。



与前几步不同的是，此步我们需采用 post 请求而不是 get 请求获取数据，且此步的反爬机制更为严格，这主要体现在两个方面：①封禁可疑 id　②给 cookie 打上禁止字段。我们的解决方式是：**①使用代理服务器 ①时常清理 cookie 并更新 headers 参数**。

在这里我们只需要获得评论数即可，即下图中的 total。



2.2.5 爬取固定歌曲的具体评论

此步和上一步采用的是相同的 API，事实上，网易对这个 API 的 Form Data 进行了加密，为了获取每一页的评论，我们要先进行解密，然后在每一次循环中更新这个参数以达到翻页的效果。如下为该步骤用到的 **AES 高级加密标准**：

```
def get_params(first_param):
    iv = "0102030405060708"
    first_key = forth_param
    second_key = 16 * 'F'
    h_encText = AES_encrypt(first_param, first_key, iv)
    h_encText = h_encText.decode('utf8')
    h_encText = AES_encrypt(h_encText, second_key, iv)
    return h_encText.decode('utf-8')


def AES_encrypt(text, key, iv):
    pad = 16 - len(text) % 16
    text = text + pad * chr(pad)
    encryptor = AES.new(key.encode('utf-8'), AES.MODE_CBC, iv.encode('utf-8'))
    encrypt_text = encryptor.encrypt(text.encode('utf-8'))
    encrypt_text = base64.b64encode(encrypt_text)
    return encrypt_text
```

在这个资源中，我们还可以得到评论的用户信息：


```
▼user: {locationInfo: null, liveInfo: null, experts: null, authStatus: 0,…}
    locationInfo: null
    liveInfo: null
    experts: null
    authStatus: 0
    avatarUrl: "https://p1.music.126.net/KoX-Va5W_VI7McUWPdGCiw==/1364493952376655.jpg"
    vipRights: null
    userId: 110761712
    userType: 0
    nickname: "智商低下的倉鼠"
    vipType: 0
    remarkName: null
    expertTags: null
```

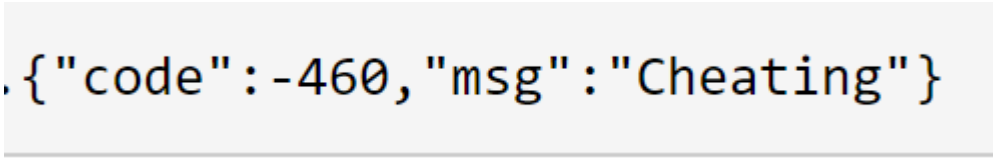

2.2.6 爬取评论的用户信息

根据上面给出的用户信息，可通过 userId 爬取用户具体的信息。通过网易云给出的官方 API 即可获得：

```
'https://music.163.com/api/v1/user/detail/'
```

从中获取用户名，生日，所在地区，性别这几个信息，这些在 API 中都已明确给出。

至此即为爬取的所有信息。其中共有的问题是会被反爬虫机制限制，从而得到以下字段：



通常可以通过 使用代理 IP 和 清除 Cookie 的方式来解决。

2.3 数据预处理

2.3.1 数据去重

①查看无重复数据：

SELECT distinct ARTIST_ID, ARTIST_NAME FROM musicdb.artists;

②查看重复的数据以及重复次数：

SELECT ARTIST_ID, COUNT (ARTIST_ID) FROM musicdb.artists GROUP BY ARTIST_ID HAVING COUNT(ARTIST_ID) > 1;

③清理重复数据:

CREATE table musicdb.albums (SELECT distinct * FROM musicdb.albums_old);

创建一个新表，将旧表中不重复数据移入即可。

2.3.2 选取评论数排行 top100 的歌曲

SELECT * FROM (SELECT * from musicdb.comments order by COMMENTS desc) as tmp limit 100;

3 数据分析及可视化

3.1 情感分析

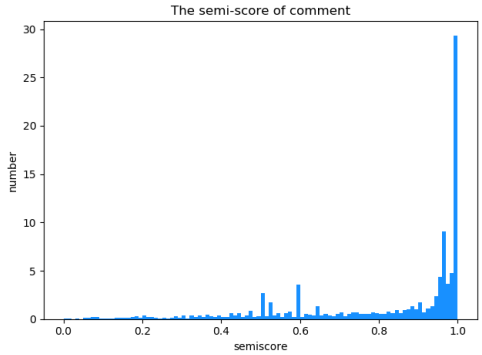
调用 snownlp 我们就可以方便地对一句话进行情感分析。我们对每一条评论做情感分析，并将值存入一个数组。

```
for line in fr.readlines():
    try:
        line = line.strip().split(',')[2]
        score.append(SnowNLP(line).sentiments)
    except:
        pass
```

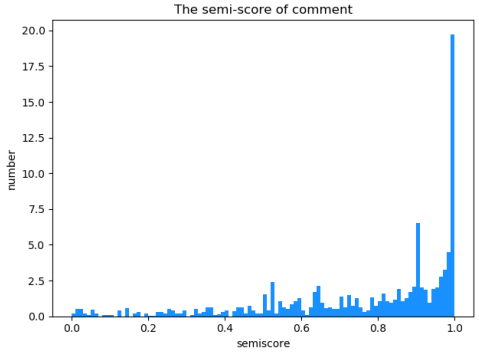
然后利用这个数组画柱状图即可。

```
plt.hist(score, bins=np.arange(0, 1.01, 0.01),normed=1,label='semisocre', color='#1890FF')
```

对《生日快乐》和《城南花已开》做情感分析图



生日快乐



城南花已开

在情感分析图中，数值越接近 1.0 表明情感越积极，反之，数值越接近 0.0 则是越消极。此处用如上两首歌来做分析，可看到《城南花已开》的情感相对《生日快乐》来说要消极一些。

3.2 Wordcloud 词云

首先使用 jieba 库对爬取到的评论信息进行分词处理。

```
comment_text = jieba.cut(''.join(text[1:]))
```

然后调用 wordcloud 库即可方便地画出词云图：

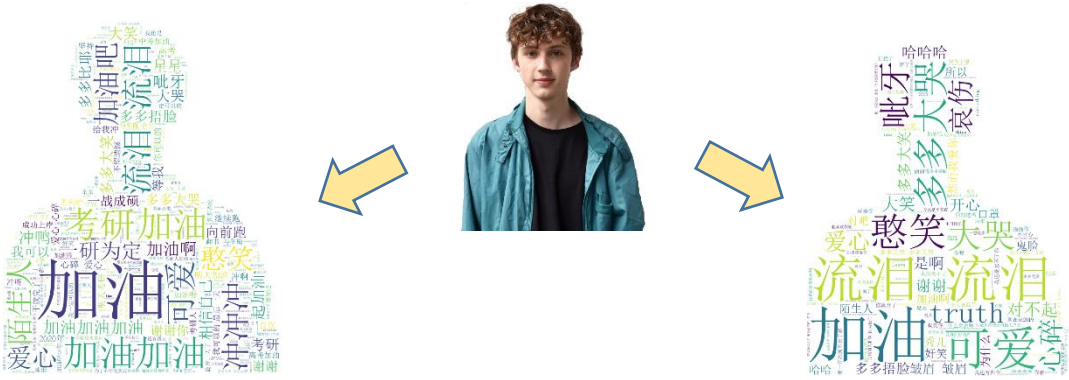
```
animal = numpy.array(Image.open('timg_meitu_3.jpg'))
wc = WordCloud(font_path='C:/Windows/Fonts/simsun.ttc', background_color="white", width=913,
               height=900,
               max_words=2000, mask=animal)
```

此处 animal 是云图的一个模板，设规定了云图大致的形状。

生日快乐
（以鸽子为模板）

城南花已开
（以企鹅为模板，但其灰色和背景白色未被区分，故成品无形状）

在词云中，出现次数越多的词句在图中显示的字体就越大，此处依旧用做情感分析的两首歌来做词云图，可以很好地看到，词云和情感分析一样，能在一定程度上反应歌曲的情感倾向。



之后，也尝试以“戳爷”为模板做了《追梦赤子心》和《the truth you leave》的词云图。

3.3 评论数随时间变化分布曲线

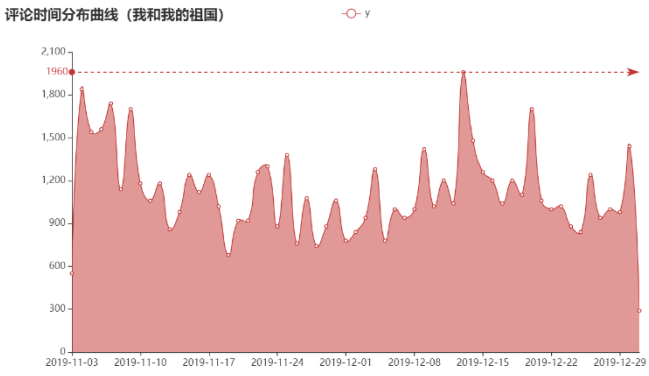
首先将网页的时间戳转换为年月日的时间，然后统计每一天的评论数即可：

```
inputs = {'id': id, 'date': date}
df = pd.DataFrame(inputs)
df['date'] = pd.to_datetime(df['date'], format='%Y-%m-%d')
df["day"] = df['date'].dt.date
curve_day = df["id"].groupby(df["day"]).count()
```

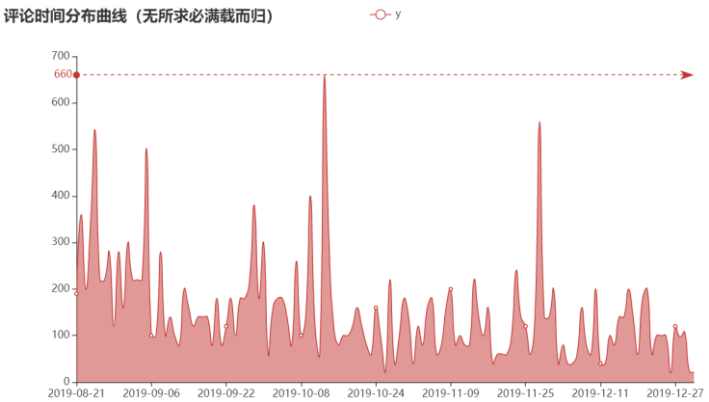
然后调用 pyecharts 画折线图：

```
line = (Line().add_xaxis(list(curve_day.index.values)).add_yaxis('y',
    .set_series_opts(
            areastyle_opts=opts.AreaStyleOpts(opacity=0.5),
            label_opts=opts.LabelOpts(is_show=False),
        ).set_global_opts(
    title_opts=opts.TitleOpts(title="评论时间分布曲线（无所求必满载而归）")
    xaxis_opts=opts.AxisOpts(
        axistick_opts=opts.AxisTickOpts(is_align_with_label=True),
        is_scale=False,
        boundary_gap=False,
    )
```

这里选取了《我和我的祖国》及《无所求必满载而归》两首歌来绘制图，可看到《无所求必满载而归》的评论数整体趋势是随时间的推后而减少，这也是大多数歌曲的热度趋势。

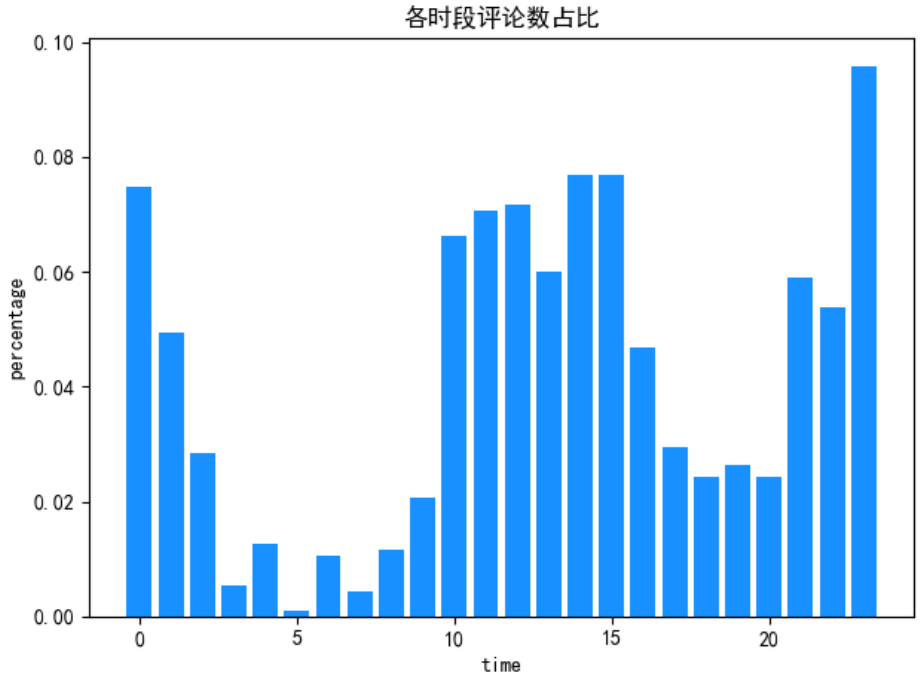


而《我和我的祖国》的评论数变化在该区间段内则相对平稳，推测是因为人们都在庆祝建国 70 周年，导致这首歌又被大家重新播放聆听，所以热度不减。可看出，时事政治对文化娱乐也会产生一定的影响。



3.4 各时段内用户评论数占比

将一天分为 24 段，并按照评论的时间在 24 小时中计数并归一化，然后调库绘出柱状图即可。



由该图分析可知，人们大多喜欢在中午及晚上听歌评论，这也很符合我们现代都市人的生活作息时间。

3.5 用户分布地区热力图

首先从网上获得地区编号与地区的对应关系，以将数据集中的地区信息转化：

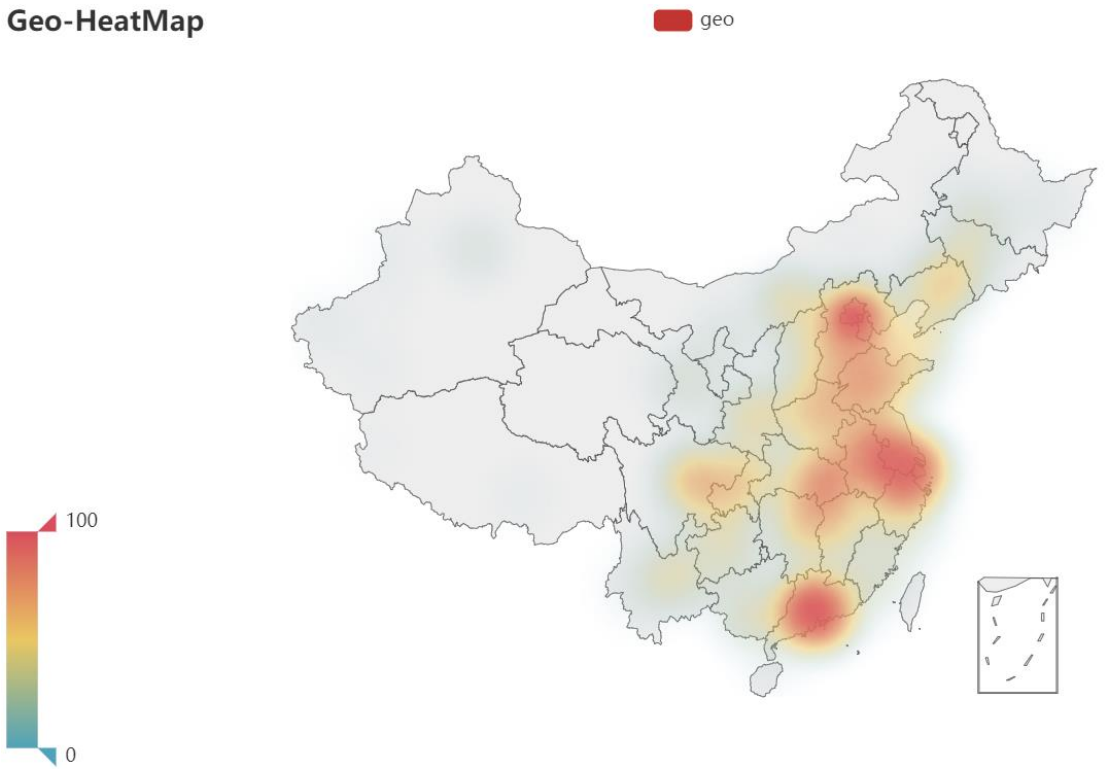
```
110000  北京市
110101  东城区
110102  西城区
110105  朝阳区
110106  丰台区
110107  石景山区
110108  海淀区
110109  门头沟区
110111  房山区
110112  通州区
110113  顺义区
110114  昌平区
110115  大兴区
```

（图示为其中一小部分）

然后按照这个关系我们可以获取到用户所在地区，对该地区的用户数量进行计数，将值放缩到 0 与 100 之间，即可调用 pyecharts 画出地图热力图：

```
for key in city.keys():
    c=[]
    c.append(key)
    c.append(city[key]*100/max_price)
    cities.append(c)
```

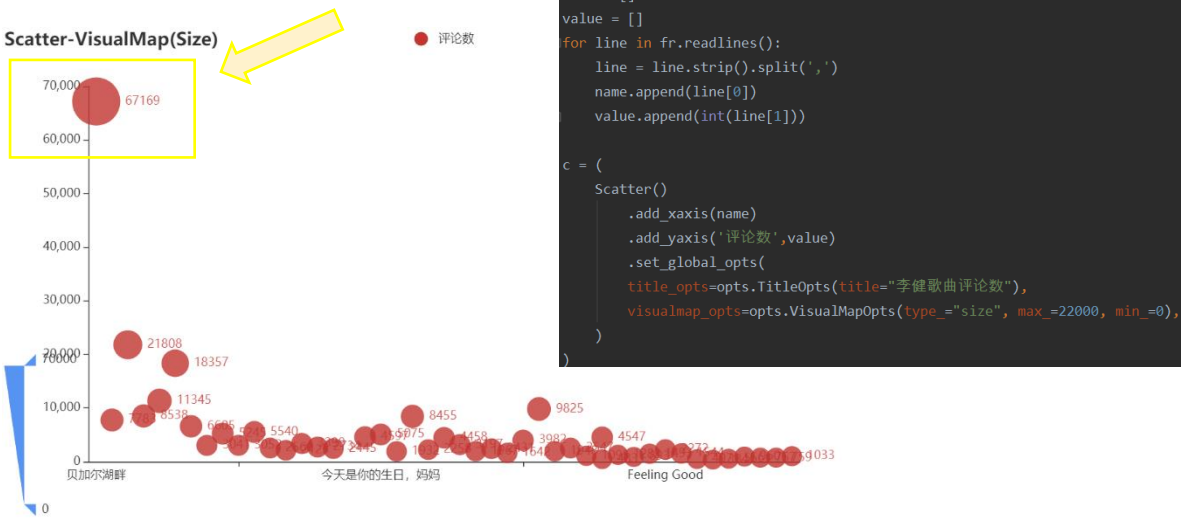
```
c = (
    Geo()
        .add_schema(maptype="china")
        .add(
        "geo",
        cities,
        type_=ChartType.HEATMAP,
    )
        .set_series_opts(label_opts=opts.LabelOpts(is_show=False))
        .set_global_opts(
        visualmap_opts=opts.VisualMapOpts(),
        title_opts=opts.TitleOpts(title="Geo-HeatMap"),
    )
)
```



上图为最终绘制出的可动图的一个截图，可见地区颜色越红，则代表越多的用户在网易云音乐上评论，可反映各地活跃用户的数量。而从图中的结果不难看出，用户活跃地区集中在北京、长三角、珠三角地区，表明文化娱乐的普及和发展也与经济状况有着密切关系。

3.6 李健歌曲评论数散点图

该步选取了本人比较喜欢的歌手李健的歌曲，调用 pyecharts 库绘制散点图。

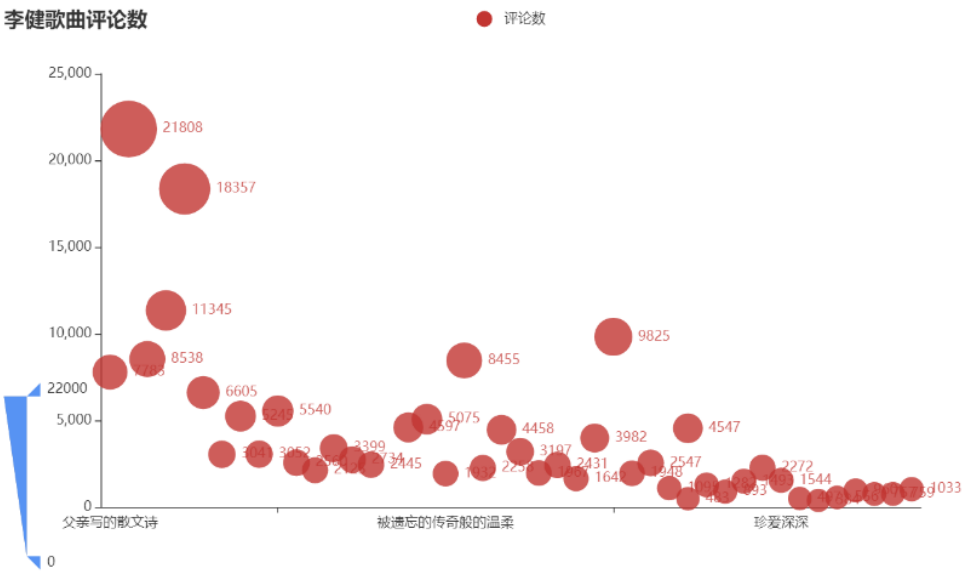



```python
fr = open('lijian.csv', encoding='utf_8_sig', errors='ignore')

name = []
value = []
for line in fr.readlines():
    line = line.strip().split(',')
    name.append(line[0])
    value.append(int(line[1]))

c = (
    Scatter()
        .add_xaxis(name)
        .add_yaxis('评论数',value)
        .set_global_opts(
        title_opts=opts.TitleOpts(title="李健歌曲评论数"),
        visualmap_opts=opts.VisualMapOpts(type_="size", max_=22000, min_=0),
    )
)
```

绘制出图后发现有一首《贝加尔湖畔》的评论数远远高于其它歌曲，导致其它歌曲评论数之间的差距看得不是特别明显。因此，又通过更改允许的最大评论数的值来去除《贝加尔湖畔》：

```python
visualmap_opts=opts.VisualMapOpts(type_="size", max_=22000, min_=0),
```

修改后的绘图如下（截图）：



通过滑动左下角的滑标可控制显示低于相对应评论数的歌曲的点，而将鼠标移至点上，还可见改点对应的歌曲名。

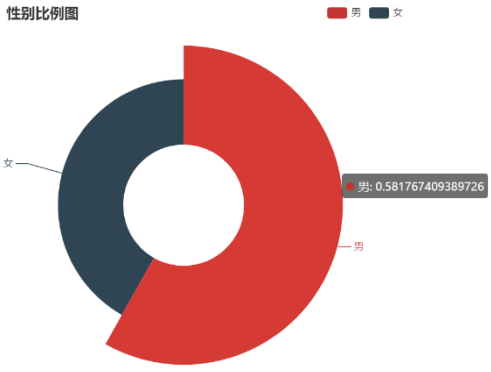
3.7 用户性别、出生比例图

同样，从用户信息中我们可得到性别和出生年份的信息，计数并归一化即可做出饼图：

①性别计数：

```
for line in fr.readlines():
    line = line.strip().split(',')
    try:
        if line[2] == '1':
            male += 1
        elif line[2] == '2':
            female += 1
    except:
        continue
```

```
e = (
    Pie()
        .add(
        "",
        [('男', perOfMale), ('女', perOfFemale)],
        radius=["30%", "75%"],
        center=["25%", "50%"],
        rosetype="radius",
    )
        .set_global_opts(title_opts=opts.TitleOpts('性别比例图'),
                         legend_opts = opts.LegendOpts(is_show=True)
                         )
)
```

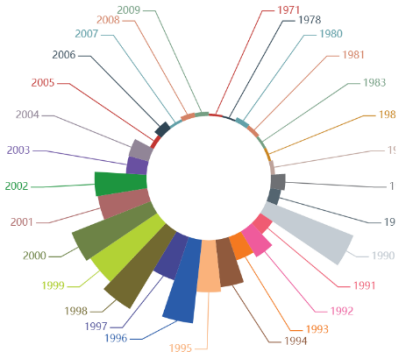


绘制出用户性别比例的玫瑰图，可见男性用户比女性多，占比达 58.18%。

②出生年份计数（忽略 2010 后出生的人）：

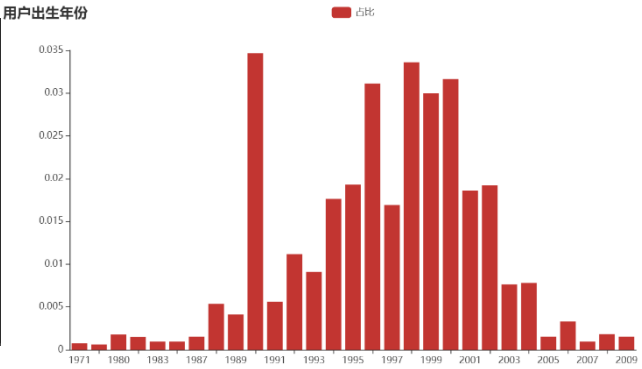
```
if line[1] != '0' and int(line[1].split(' ')[0].split('-')[0])<2010:
    try:
        year[line[1].split(' ')[0].split('-')[0]] += 1
    except:
        year[line[1].split(' ')[0].split('-')[0]] = 1
```

```
c = (
    Pie()
        .add(
        "",
        years,
        radius=["30%", "75%"],
        center=["75%", "50%"],
        rosetype="area",
    )
        .set_global_opts(title_opts=opts.TitleOpts('出生年份比例图'),
                         legend_opts=opts.LegendOpts(is_show=False)
                         )
)
```



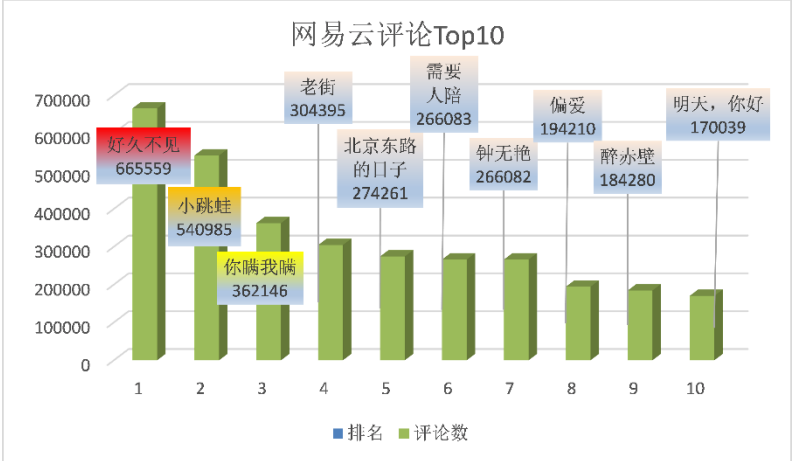
为了防止饼图大小不直观，我们同样绘制了柱状图：

```
d = (
    Bar(
        init_opts=opts.InitOpts(
            animation_opts=opts.AnimationOpts(
                animation_delay=500, animation_easing="elasticOut"
            )
        )
    )
    .add_xaxis(years1)
    .add_yaxis('占比',values)
    .set_series_opts(
        label_opts=opts.LabelOpts(is_show=False),
    )
    .set_global_opts(
        title_opts=opts.TitleOpts(title='用户出生年份')

    )
)
```



3.8 评论数 top10 三维柱状图

选取在 MySQL 中操作获得的评论数 top10 的歌曲信息，利用 Excel 绘制三维柱状图。



4 后期展望

在做该项目时，还有如下一些想法没来得及实现，期望后期能够完善并实现它们。

① 通过用户在歌手的歌曲下方评论，以及评论区用户之间的回复互动等，建立关系网络。

② 对所做绘制的用户分布地区热力图进行完善，最终以动图的方式呈现，可展示一天时间之内，在各个时间段内各地用户评论多少的情况。（类似夜间灯火图，有用户评论则用户所在地区发光，评论的用户越多，光越亮。）