

Python 课程设计文档

豆瓣电影 TOP250 数据分析

171860526 褚有刚 171860029 张雨

2020 年 6 月 10 日

1 概述

1.1 背景

豆瓣(douban)是一家中国社交网站,以书影音起家,亦是一个在线数据库,提供关于书籍、电影、电视、音乐、游戏、舞台剧等作品的信息,无论描述还是评论都由用户提供,是中国 Web 2.0 网站中具有特色的一个网站。网站还提供书影音推荐、线下同城活动、小组话题交流等多种服务功能,它更像一个集品味系统(读书、电影、电视、音乐)、表达系统(我读、我看、我听)和交流系统(同城、小组、友邻)于一体的创新网络服务,一直致力于帮助都市人群发现生活中有用的事物。

1.2 课程设计目标

对豆瓣电影榜单 TOP250 进行数据分析。整个过程包括爬取网页信息,将爬取的数据存储到数据库,对数据进行分析,和使用网页展示数据分析结果几个部分。

1.3 小组成员

171860526	计算机科学与技术	褚有刚
171860029	计算机科学与技术	张雨

1.4 成员分工

数据爬取	褚有刚	
数据分析	褚有刚	张雨
结果展示	褚有刚	张雨
实验报告	褚有刚	张雨

1.5 代码目录结构

- db: 存储 sqlite 数据库以及相应的建表语句;

- movie: scrapy 框架实现的爬虫功能;
- static: 网页的静态资源;
- template: 网页的 html 文档;
- wc: 生成词云图片;
- analysis.py: 生成数据分析的一些静态图片;
- app.py: 程序的入口。

1.6 代码运行方式

运行 app.py 然后使用浏览器访问 localhost:5000

2 数据获取

2.1 网页内容分析

通过浏览器查看, 各部电影的信息主要分布在电影的详情页, 可以通过豆瓣 Top250 的主页跳转。除此之外, 分析一部电影的内容还需要使用到他们的评论, 各部电影的评价页可以通过电影的详情页的跳转。



2.2 爬取数据

使用 scrapy 框架, 通过网络爬虫的方式获取, 其中 Scrapy 是 Python 开发的一个快速、高层次的 web 抓取框架。具体的步骤如下。

首先从豆瓣 Top250 主页开始, 爬取各个电影的详情页的超链接。

然后从各个电影的详情页爬取电影的导演, 主演, 类型等信息, 以及评论页的超链接, 其中评论只从各个电影的评论页爬取前 20 条热评, 因为过多的评论会导致数据集过大, 并且前 20 条热评已经具有代表性。

2.3 部分代码实现

```
class DoubanSpider(CrawlSpider):
    name = 'douban'
    allowed_domains = ['movie.douban.com']
    start_urls = ['http://movie.douban.com/top250']

    rules = (
        Rule(LinkExtractor(allow=r'/top250.*'), callback=None, follow=True),
        Rule(LinkExtractor(allow=r'/subject/\d+/$'), callback='parse_movie', follow=True),
        Rule(LinkExtractor(allow=r'/subject/\d+/comments/?status=P$'), callback='parse_comment', follow=False),
    )

    def parse_movie(self, response: Response):
        item = {}
        item['entity'] = 'movie'
        item['movie'] = response.xpath('//hl/span[@property="v:itemreviewed"]/text()').get().split()[0]
        item['year'] = response.xpath('//hl/span[@class="year"]/text()').get()[1:-1]
        item['score'] = response.xpath('//strong/text()').get()
        item['director'] = response.xpath('//a[@rel="v:directedBy"]/text()').getall()
        item['actor'] = response.xpath('//a[@rel="v:starring"]/text()').getall()
        item['genre'] = response.xpath('//span[@property="v:genre"]/text()').getall()
        info = ''.join(response.xpath('//div[@id="info"]/text()').getall())
        item['country'] = info.replace('/', '').split()[0]
        item['length'] = re.search(r'\d+', response.xpath('//span[@property="v:runtime"]/text()').get()).group()
        item['rank'] = re.search(r'\d+', response.xpath('//span[@class="top250-no"]/text()').get()).group()
        item['img_url'] = response.xpath('//div[@id="mainpic"]//img/@src').get()
        return item

    def parse_comment(self, response):
        item = {}
        item['entity'] = 'comment'
        item['movie'] = response.xpath('//hl/text()').get().split()[0]
        item['comment'] = response.xpath('//span[@class="short"]/text()').getall()
        return item
```

3 数据存储

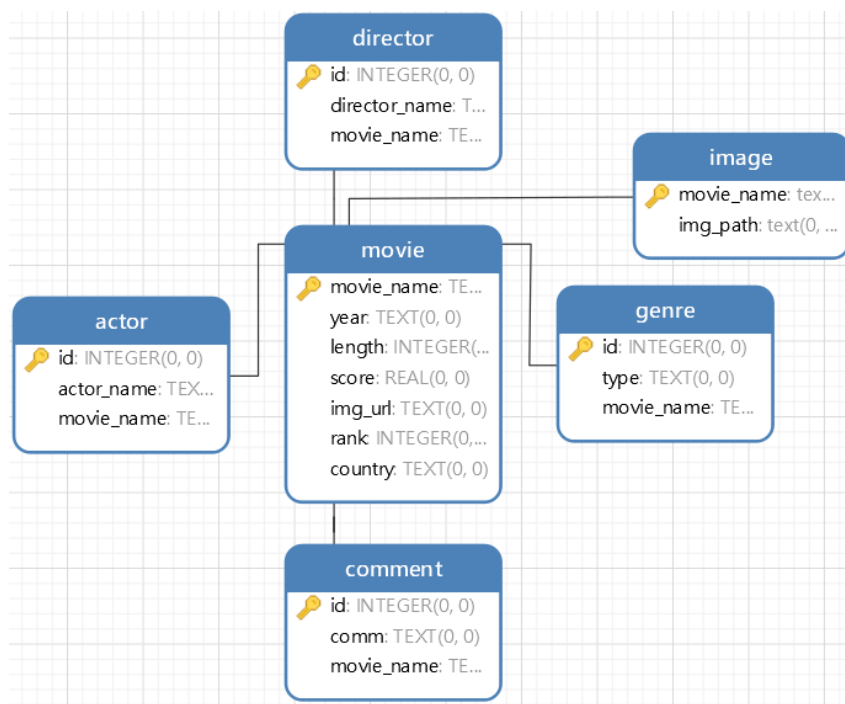
3.1 数据表结构设计

将前面获取到的数据存储到数据库中，方便之后对数据进行分析。

在设计数据表结构时，同时考虑到之后进行数据分析的方便性以及规范化原则，设计了电影，导演，主演，类型，评论，图片共六个数据表。

核心为的 movie 表。包含主键 movie_name，它为 TEXT 类型，movie 表包含属性上映年份，电影时长，电影评分，电影海报链接，电影排名，电影国家/地区。其他的数据表均有外键 movie_name，他们都引用了 movie 表中的电影名称这个主键，从而可以通过表连接的方式查询更多的相应数据。

3.2 数据表结构设计图



3.3 部分代码实现

```
def process_movie(self, item):
    sql = f'''INSERT INTO movie
(movie_name, year, length, score, img_url, rank, country)
VALUES
(' {item['movie']} ', ' {item['year']} ', ' {item['length']} ', ' {item['score']} ',
' {item['img_url']} ', ' {item['rank']} ', ' {item['country']} ')'''
    self.cursor.execute(sql)
```

4 数据分析

4.1 数据分析思路

由于通过爬虫获取数据之后将数据存储数据库中，因此可以使用 SQL 进行一些简单的初步分析，然后转换为 DataFrame 数据结构再使用 pandas 进行进一步的分析，对于分析的结果通过静态图片或者 web 页面的方式展示。

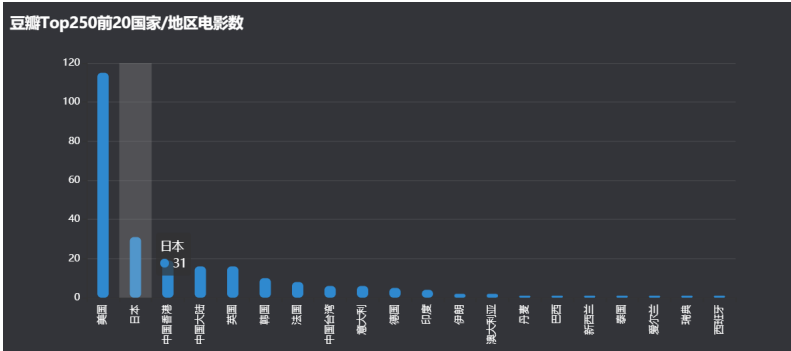
如下图，是 SQL 语句的使用实例，使用 select 语句获取相应的数据，然后通过网页前后端的数据交互从而显示在网页中。

```
with sqlite3.connect('db/movie.db') as conn:
    country = pd.read_sql(
        'select country, count(*) as cnt from movie group by country order by cnt desc limit 20', conn)
    country = {'x': list(country.country), 'y': list(country.cnt)}
```

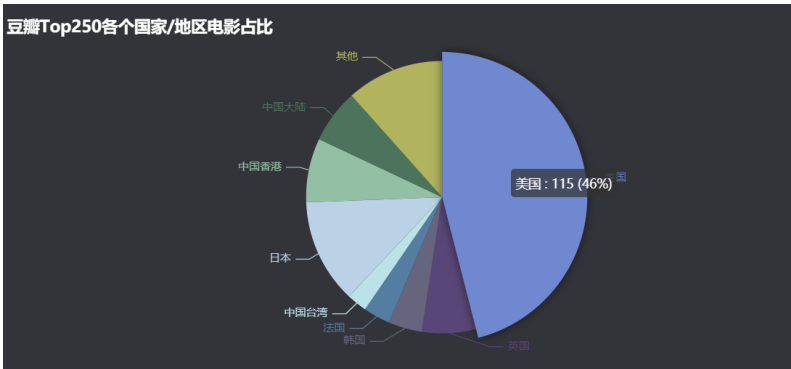
4.2 国家/地区数据分析

在豆瓣 Top250 的电影中，美国出产的电影的占比一骑绝尘，领先了其他国家一大截。为什么全世界只有美国产生了这样体量级的电影工业，原因是多样的。因为美国有全球发行的能力，想要保持真正的全球发行能力，就需要对全球各个市场的有着同等高度的理解力，并且能够在同一部电影里去平衡不同文化差异所带来的理解偏差，让全球人民都认同同一种观念。另外美国创意人才面向全世界开放，美国有更成熟的商业机制，美国特效工业的技术能力更强等等原因。

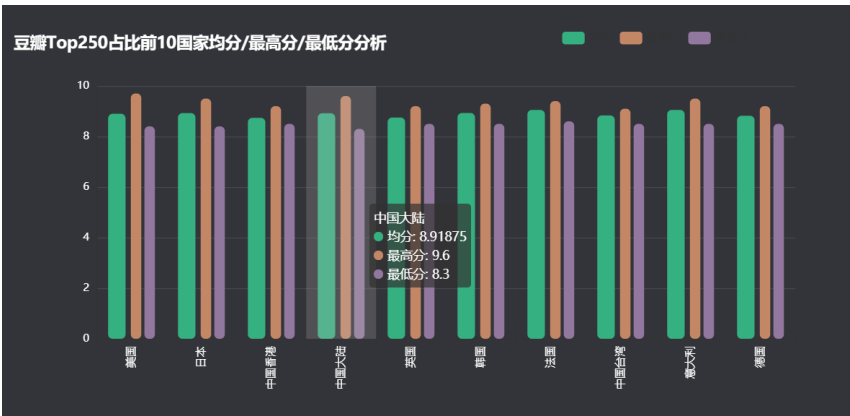
由于来自中国的电影是分为大陆，香港，台湾三部分统计的，如果将这些电影放在一起统计，中国将能够排到第二名，超过原本的第二名日本约 10 名左右，可见表明中国的电影发展有很大的发展潜力。



前 6 名的国家，包括美国，中国，日本，英国，韩国，法国的电影数合计占据前 250 榜单的 88.4%，即优质电影集中来自于少部分国家。

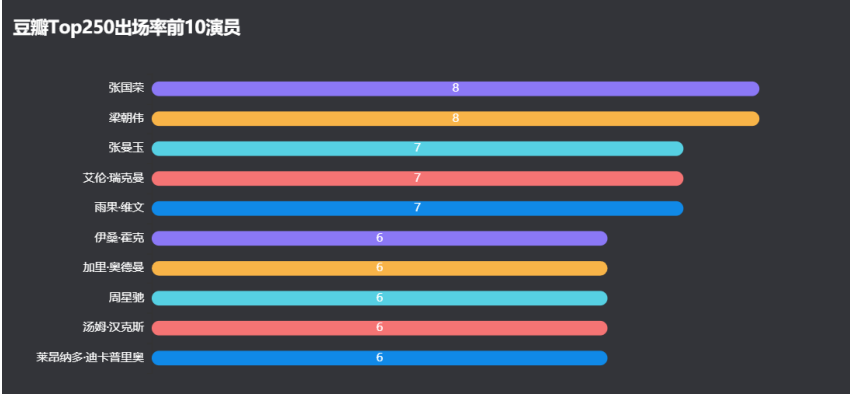


占比前 10 的国家，排名越前，如美国/中国大陆，因为电影数量多，最高与最低分的差距也相对更大，但是各个国家的均分都在差不多的水平。

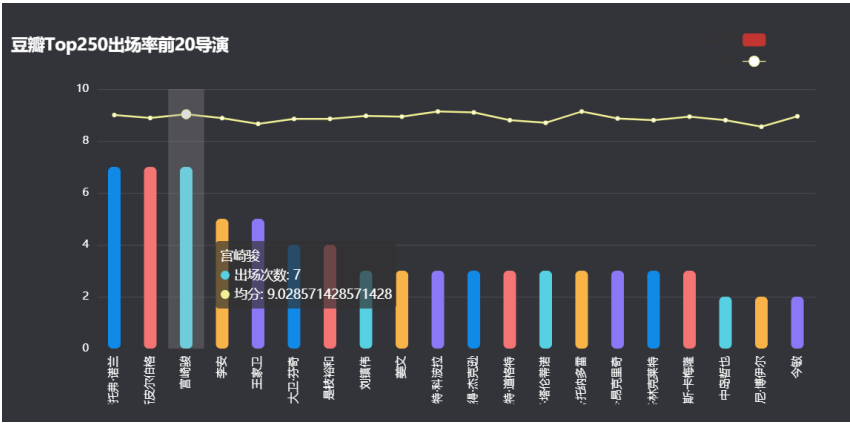


4.3 演员/导演数据分析

前 10 演员都是十分著名的国际演员，可以看到，其中中国的演员占据了 4 位，分别是张国荣/梁朝伟/张曼玉/周星驰，均为来自中国香港 90 年代知名演员，仔细思考就会发现，在 Top250 中单是香港的电影数就排到了第三位，而香港娱乐圈人又不是很多，因此香港的艺人出演 Top250 的电影十分多也就不足为奇了。同时由于豆瓣是中文网站，所以中国演员的占比很高也在情理之中。



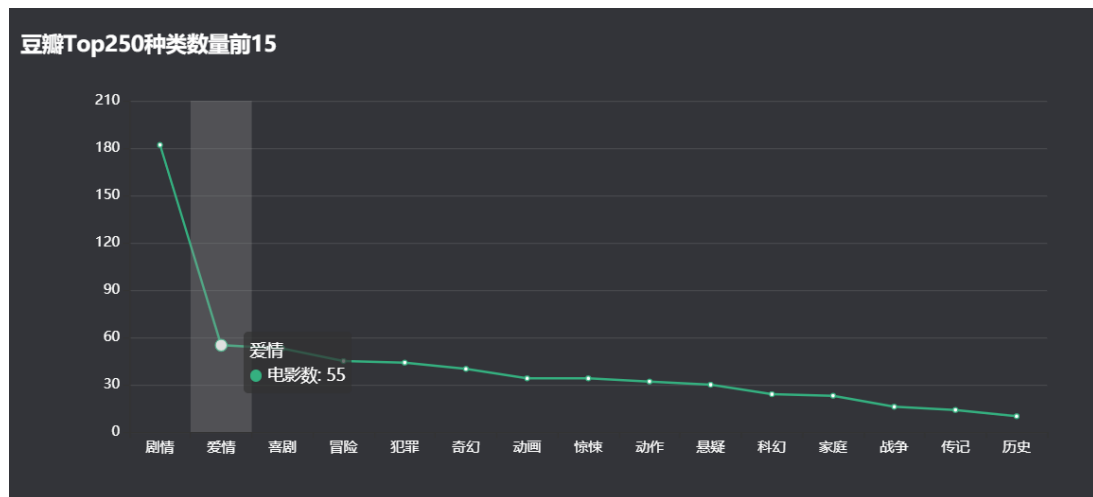
在导演排名中，也不乏有很多熟知的导演，如排名第一的托弗诺兰，他执导的《盗梦空间》《星际穿越》《敦刻尔克》《致命魔术》都有很高的国际认可度，并列第一的宫崎骏也为大家熟知。



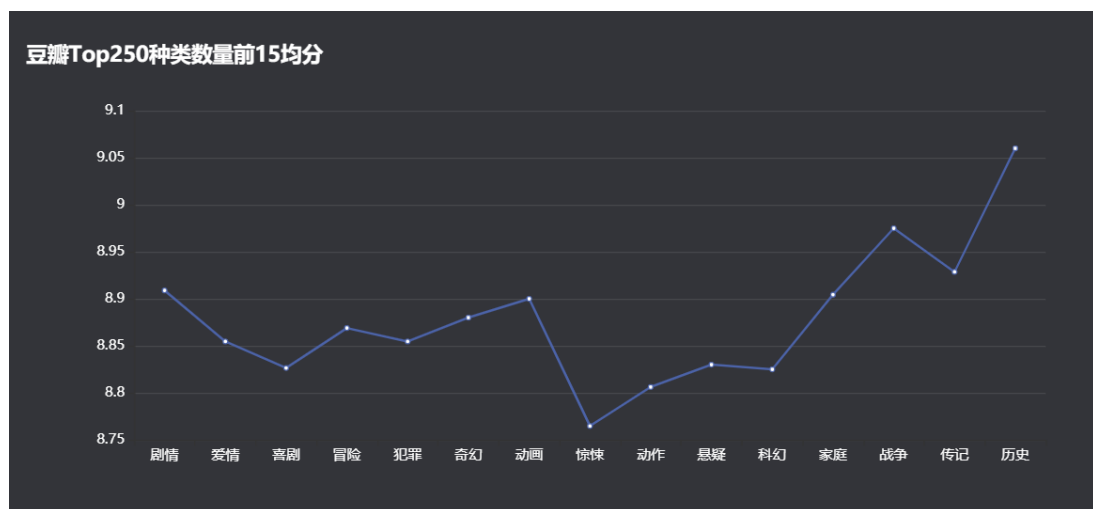
4.4 电影类型数据分析

电影类型主要有动作、冒险、动画、传记、喜剧、犯罪、剧情、家庭、奇幻、历史、恐怖、音乐、歌舞、悬疑、爱情、科幻、体育、惊悚、战争和西部电影等等。

可以看到剧情片占 250 部电影中的 182 部，排名第二的为爱情片为 55 部，剧情片的数量是第二名的三倍多。剧情片为何能够成为电影制作的主要类型？主要原因有剧情片成本低，因为电影不需要特别的布景、服装、地点、道具、视觉特效等等。并且剧情在所有类型的电影中具备最广泛的定义，因为任何地方发生的一切事情都可以是剧情。相反，其他类型的电影却有更清晰的分类标准，比如将展现激烈冲突的电影归类为动作片，而讲述惊悚事件的电影是恐怖片，具备搞笑元素的电影则是喜剧片等等。

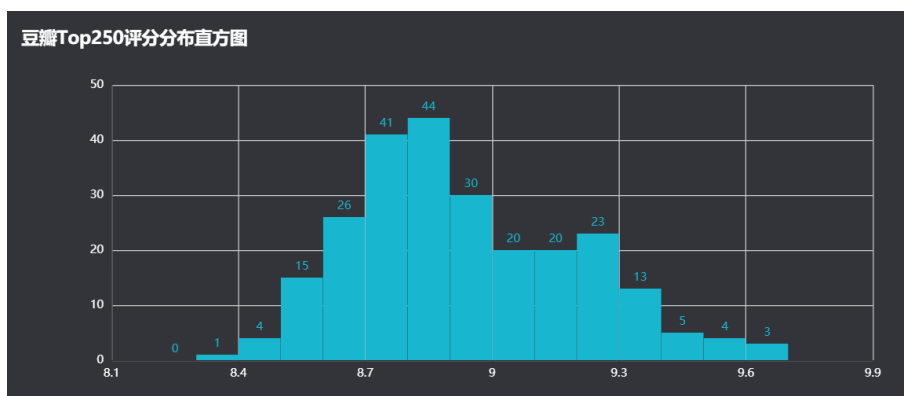


在各类型的电影评分均分中，最低的是惊悚片，最高的是历史片，虽然仅有 10 部在榜单。

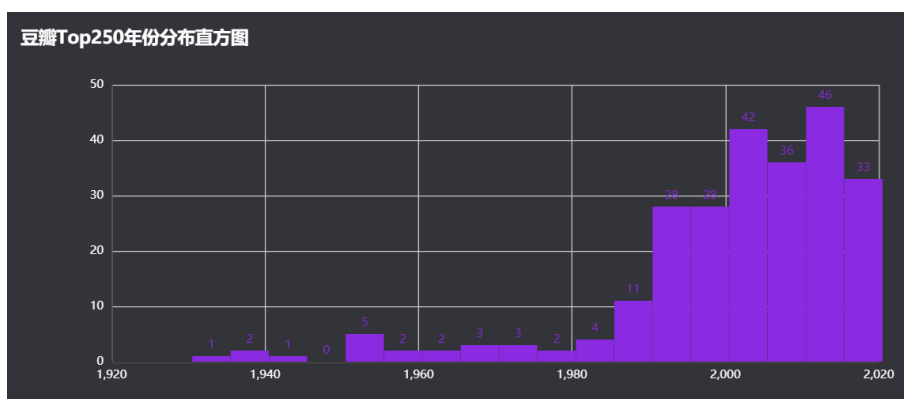


4.5 电影年份数据分析

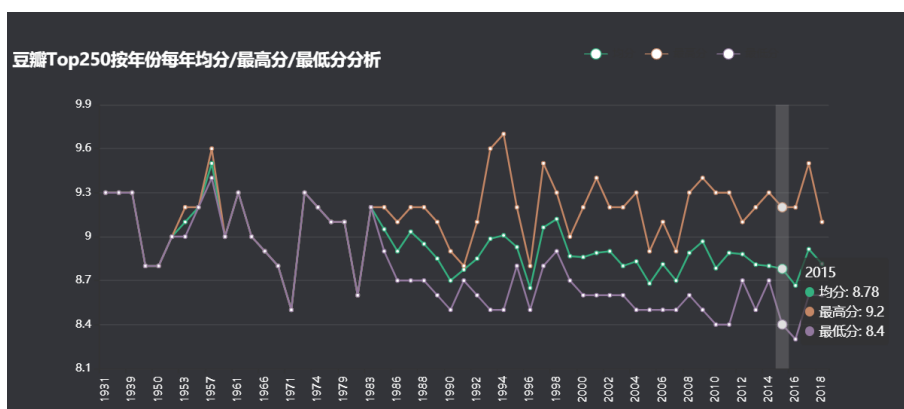
首先下面是 TOP250 整体的评分分布直方图，高分电影在 8.7-9.0 区间最多。高于 9.3 分的高分区间的分数相对较小，基本上服从了常见的正态分布。



电影数随着年份的增长呈现出指数增长的趋势，这主要是在 20 世纪，绝大多数国家刚刚经历战争，经济发展才刚刚起步，电影工业还不是很发达，而进入 21 世纪以来，大众的娱乐生活逐渐丰富，电影走进了千家万户，商业开始逐渐转向电影投资，好电影也都如同雨后春笋般冒了出来。



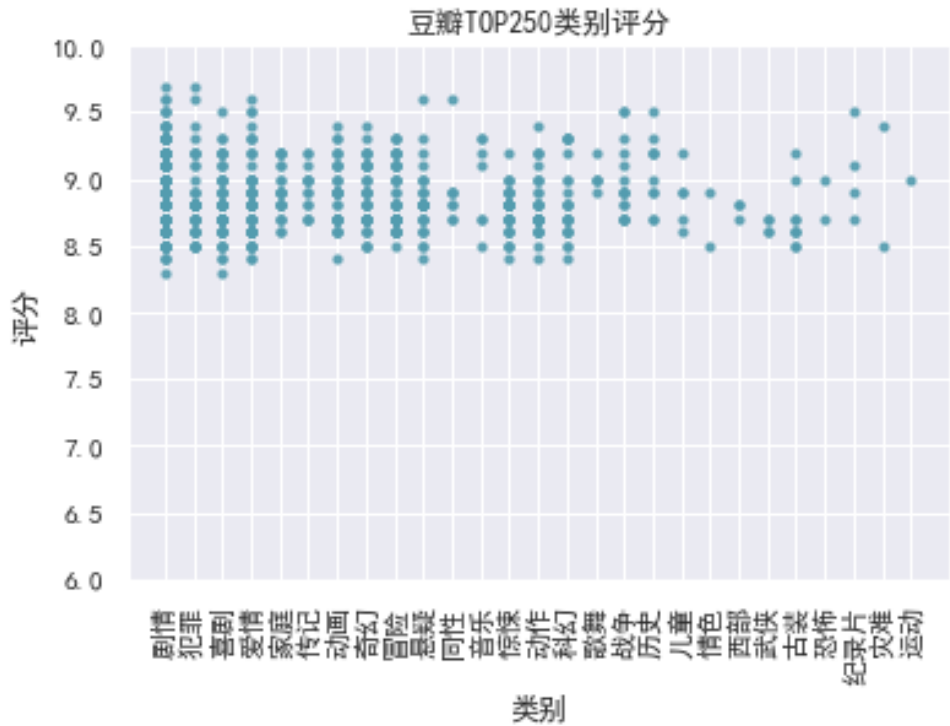
Top250 中绝大多数电影都集中在近些年，在早些年，每年可能只有一部或者几部电影，因此评分的差异化很小，而在近些年，涌现了大批的好电影，因此电影的评分差异化开始增加，最高分和最低分的评分之间的差距拉大，但是总体来说，评分都保持在比较高的水平。



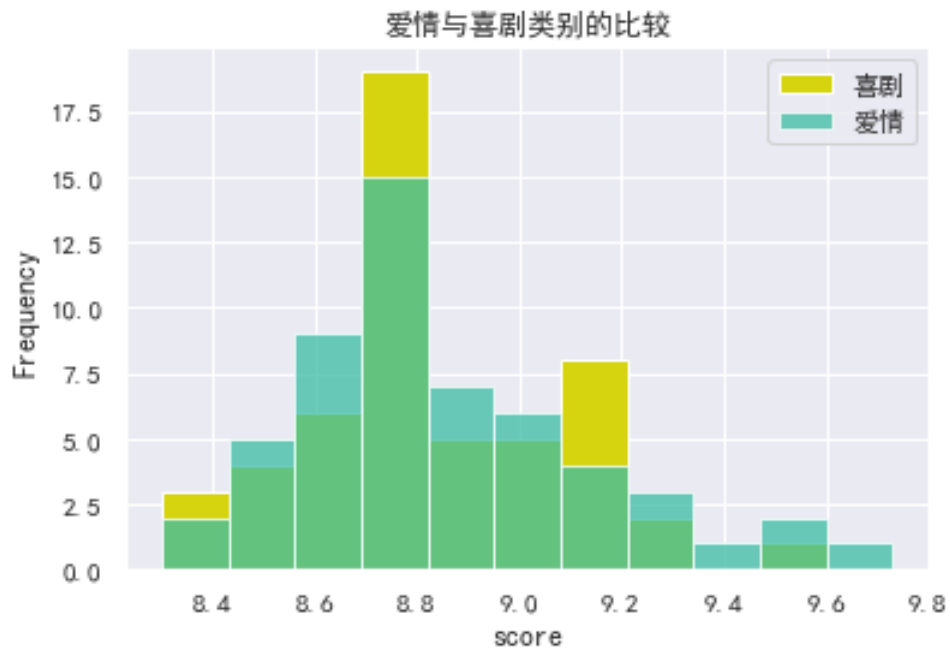
4.6 其他数据分析

最后是一些其他的数据分析。这里是希望通过不同的图表类型再次审视数据。

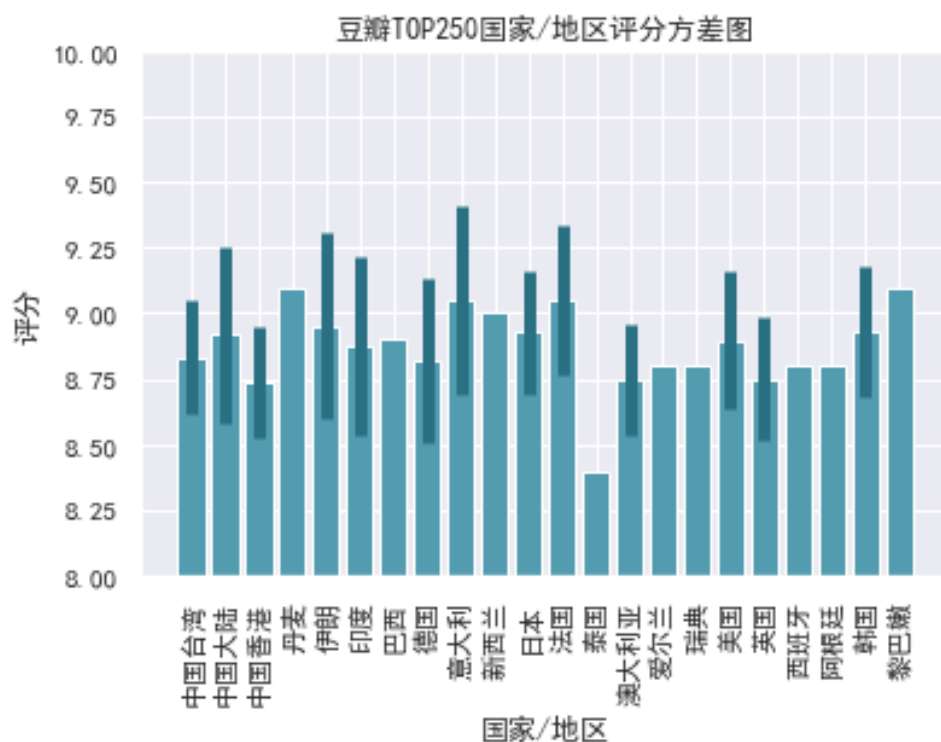
如下是散点图，散点图给人更形象化的表示，从另一种图表形式反映了电影类型与对应的上榜数量，剧情片最多，同时包括评分分布，它从三个维度反映了数据。



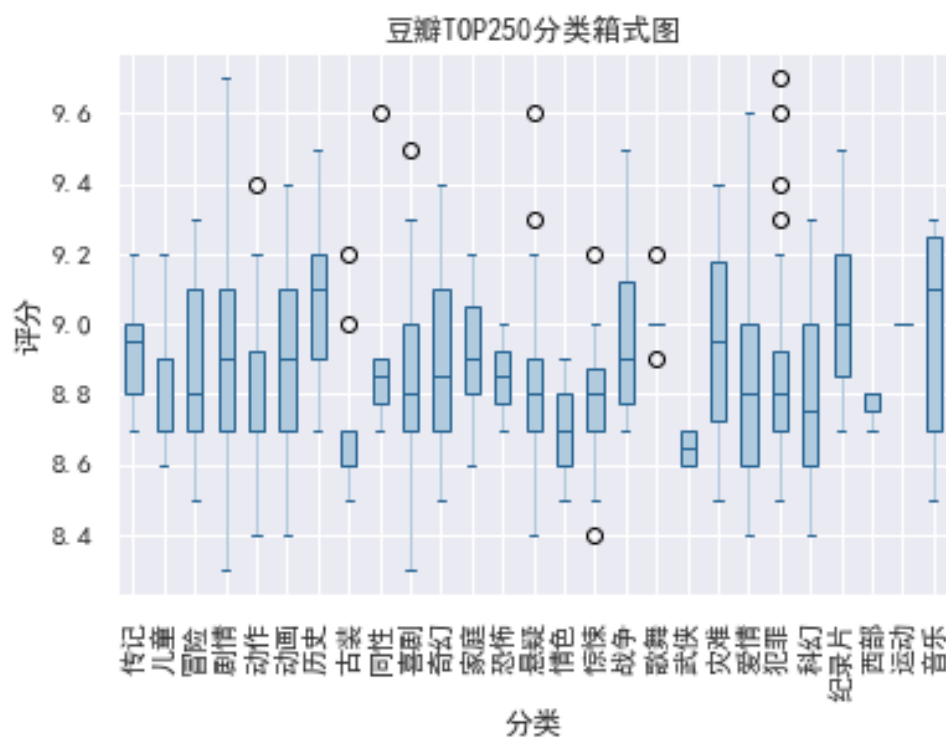
如下是叠柱状图。在前面的统计中，爱情电影有 55 部，戏剧电影有 53 部。所以将上榜数量较为相似的二者进行比较。整体来看，爱情电影的评分分布更均匀，喜剧电影的评分更集中在中间区域。



如下是均值方差图。柱状是均分，细线是方差，表现了地区评分的离散程度。这里因为有些国家上榜电影数较少，所以数据并没有很大的参考意义，但是均值方差图在很多场合都适用。



最后是箱式图，散点是异常值，图中给出了中位数，四分位数等，这里同样能看出历史的评分中位数更高。



4.7 词云处理

从数据库中读取评论数据表，构建图云，并将图片存储。

```
with sqlite3.connect('../db/movie.db') as conn:
    df = pd.read_sql('select * from comment', conn)
    cursor = conn.cursor()
    ser = df.groupby('movie_name').comm.apply(' '.join)
    wc = WordCloud(font_path='msyh.ttc')
    for index in ser.index:
        wc.generate_from_text(' '.join(jieba.cut(ser[index])))
        wc.to_file('../static/img/wc/' + index + '.png')
        sql = f'''INSERT INTO image(movie_name, img_path) VALUES
        ('{index}', '{'/static/img/wc/' + index + '.png'})'''
        cursor.execute(sql)
    conn.commit()
    cursor.close()
```

5 可视化展示

5.1 网页首页


















5.2 豆瓣 TOP250 信息展示

可以看到每部电影的电影海报，电影名称，上映年份，片长，制作国家或地区，评分。并且可以查看下一页，上一页，回到顶部。

Home Data Analysis Word Cloud

豆瓣电影 TOP 250

电影海报	电影名称	上映年份	片长	制片国家/地区	评分
	肖申克的救赎	1994	142	美国	9.7
	教父	1972	175	美国	9.6
	教父2	1974	175	美国	9.6
	教父3	1978	175	美国	9.5
	肖申克的救赎	1994	142	美国	9.5
	美国丽人	1999	116	意大利	9.5
	教父	1972	175	美国	9.4
	千与千寻	2001	125	日本	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4

	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4
	肖申克的救赎	1994	142	美国	9.4

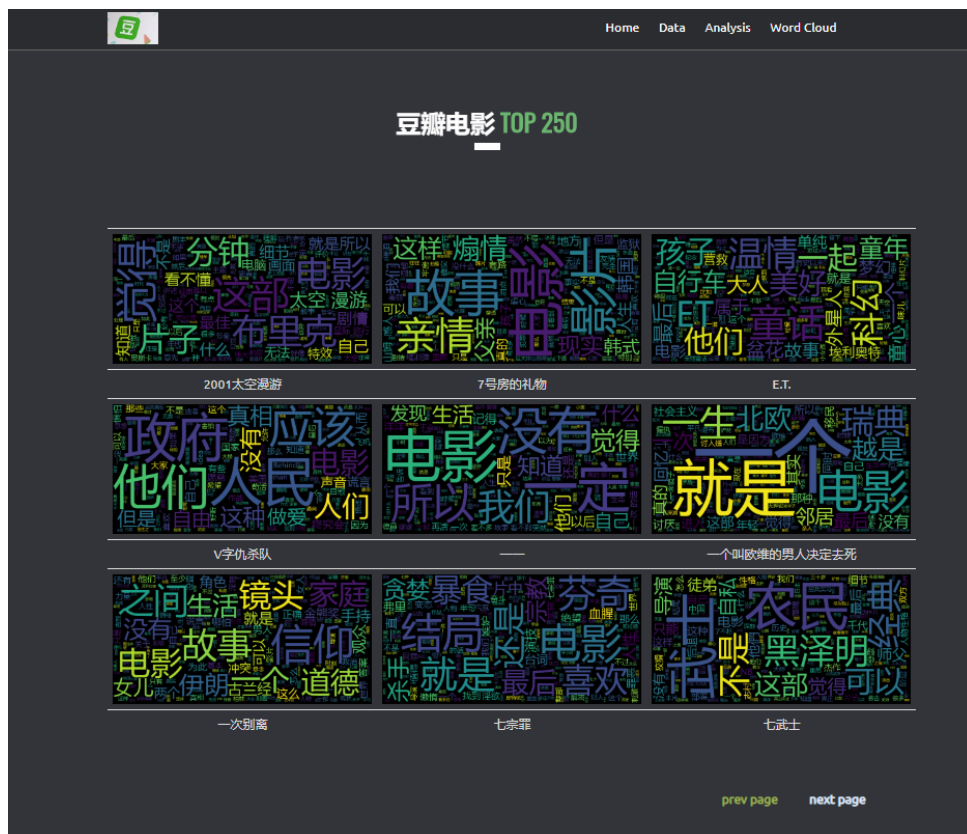
prev page next page

5.3 数据分析页面展示

有各种类型的图表，使用了 echarts，有更好的可视化效果。



5.4 词云展示



6 网页实现

6.1 flask 框架

Flask 是一个使用 Python 编写的轻量级 Web 应用框架。基于 Werkzeug WSGI 工具箱和 Jinja2 模板引擎。Flask 使用 BSD 授权。Flask 被称为“microframework”，因为它使用简单的核心，用 extension 增加其他功能。Flask 没有默认使用的数据库、窗体验证工具。然而，Flask 保留了扩增的弹性，可以用 Flask-extension 加入这些功能：ORM、窗体验证工具、文件上传、各种开放式身份验证技术。

6.2 界面跳转

以导航栏为例，将 Home/Data/Analysis/Word Cloud 链接到相应的网页。具体的 html 语句如下。

```
<nav class="collapse navbar-collapse navbar-right" role="Navigation">
  <ul id="nav" class="nav navbar-nav">
    <li><a href="/">Home</a></li>
    <li><a href="/data">Data</a></li>
    <li><a href="/analysis">Analysis</a></li>
    <li><a href="/wc">Word Cloud</a></li>
  </ul>
</nav>
```

6.3 web 功能实现

使用 js 实现网页的折叠和展开：

```
<script type="text/javascript">
  function show1() {
    var obj_div = document.getElementById("block1");
    obj_div.style.display = (obj_div.style.display=="none")?'block':'none';
  }
</script>
```

6.4 图表绘制

在图标绘制中采用了百度开源的 echarts，本来打算用 pyecharts 实现的，但是该模块的功能不够完善，因此选择通过 js 使用原生的 echarts 实现。

以“豆瓣 Top250 占比前 10 国家均分/最高分/最低分分析”为例分析，如下。

color 为图表基色；title 定义图表名称；tooltip 为提示框，即鼠标停留图表处的信息显示；legend 为图例，可以控制图例的显示，并且点击相应图例，可以控制相应部分数据的显示；xAxis/yAxis 为横纵坐标的信息控制；series 部分确定了各部分的数据，显示出的图表类型等等。

```
<script type="text/javascript">
    // 基于准备好的dom，初始化echarts实例
    var myChart = echarts.init(document.getElementById('country_rank'))
    // 指定图表的配置项和数据
    var option = {
        color: ['#2f89cf'],
        title: {text: '豆瓣Top250占比前10国家均分/最高分/最低分分析'...},
        tooltip: {trigger: 'axis'...},
        legend: {orient: 'horizontal'...},
        xAxis: {type: 'category'...},
        yAxis: {type: 'value'...},
        series: [...]
    };
    // 使用刚指定的配置项和数据显示图表。
    myChart.setOption(option);
</script>
```

6.5 前后端数据交互

通过 python 从数据库中获取所需的数据之后，通过 flask 框架的 jinja 模块进行解析，转换为 html 静态页面，从而在浏览器上呈现。

7 总结

- 1、进一步掌握了数据分析的方法，能够自主对数据进行获取并思考。
- 2、学会了部分前端书写的技能，对更多的前端框架有了了解。
- 3、能够团队合作，更好的完成任务。
- 4、对电影的有进一步的了解。