Dase 导论实验报告

数据科学与大数据技术 10225501458 周向子健

一、作品简述：

我的作品主要是通过爬取豆瓣电影 Top250 部电影的相关信息数据，包括：排名，电影名称，上映年份，国家，评分，评论人数，还有电影类型。然后主要通过 dataease 平台完成数据可视化部分。对于这个作品，我的初衷是想从这些数据找出如今电影的发展趋势，尤其是豆瓣作为一个主要有中国影民打分评论的平台，这些数据应当可以反映一些中国的大众电影审美。

二、代码部分

首先是爬虫部分：

```python
import requests
import re
from bs4 import BeautifulSoup
import pandas as pd

# 用于应对反爬机制，作为请求头
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT10.0; Win64; x64)AppleWebKit/537.36(KHTML, '
                  'like Gecko)Chrome/120.0.0.0Safari/537.36 Edg/120.0.0.0'
}

# 构造分页数字列表
page_indexes = range(0, 250, 25)


# 用于下载所有页面的内容，一共十个页面
1 usage  ▲ Zhouxzj
def download_all_htmls():
    webs = []
    for index in page_indexes:
        url = f"https://movie.douban.com/top250?start={index}&filter="
        # {index}用于分页查找的定位功能
        # print("crawl html:", url)
        r = requests.get(url, headers=headers)
        # 验证网页是否被成功爬取，r.status_code为200即证明爬取成功
        if r.status_code != 200:
            raise Exception("error")
        webs.append(r.text)
        # 将网页加入列表webs
    return webs
```

在这一段中首先就是下载豆瓣电影 Top250 的网页内容，以便后续进行清洗和提取数据。由于相关的网页不止一个，有十个分页，所以根据其地址构造了 url 便于爬取所以十个网页的内容。

```python
htmls = download_all_htmls()


1 usage  ± Zhouxzj *
def parse_single_html(html_text):
    # 解析单个网页的数据
    soup = BeautifulSoup(html_text, features='html.parser')
    # 下面是各种定位找要爬取的信息，通过html的标签进行查找
    article_items = (
        soup.find(name="div", class_="article")
        .find("ol", class_="grid_view")
        .find_all(name="div", class_="item")
    )
    datas = []
    for article_item in article_items:
        rank = article_item.find("div", class_="pic").find("em").get_text()
        info = article_item.find("div", class_="info")
        title = info.find("div", class_="hd").find("span", class_="title").get_text()
        OtherInfo = info.find("div", class_="bd").find("p", class_="").get_text()
        stars = (
            info.find("div", class_="bd")
            .find("div", class_="star")
            .find_all("span")
        )
        rating_star = stars[0]["class"]
        # 获取了span标签的class的值
        rating_num = stars[1].get_text()
        comments = stars[3].get_text()
        years = re.search(pattern=r"\d{4}", string=OtherInfo).group()
```

第二部分便是解析单个网页的数据，爬取我们需要的数据内容，首先根据页面源代码内容，我们可以定位到我们需要的数据所处的位置，如图是网页源代码：

```
ol class="grid_view">
        <li>
            <div class="item">
                <div class="pic">
                    <em class="">1</em>
                    <a href="https://movie.douban.com/subject/1292052/">
                        <img width="100" alt="肖申克的救赎" src="https://img3.doubanio.com/view/photo/s_ratio_poster/public/p480747492.webp" class="">
                    </a>
                </div>
                <div class="info">
                    <div class="hd">
                        <a href="https://movie.douban.com/subject/1292052/" class="">
                            <span class="title">肖申克的救赎</span>
                                <span class="title"> / The Shawshank Redemption</span>
                            <span class="other"> / 月黑高飞(港)  /  刺激1995(台)</span>
                        </a>


                            <span class="playable">[可播放]</span>
                    </div>
                    <div class="bd">
                        <p class="">
                            导演: 弗兰克·德拉邦特 Frank Darabont   主演: 蒂姆·罗宾斯 Tim Robbins /...<br>
                            1994 / 美国 / 犯罪 剧情
                        </p>


                        <div class="star">
                                <span class="rating5-t"></span>
                                <span class="rating_num" property="v:average">9.7</span>
                                <span property="v:best" content="10.0"></span>
                                <span>2972850人评价</span>
                        </div>

                            <p class="quote">
                                <span class="inq">希望让人自由。</span>
                            </p>
                    </div>
                </div>
            </div>
        </li>
        <li>
```

由此图我们可以发现 rank 信息在 pic 中，而其他信息均在 info 中，然后只要依每个 div 块获得相应的数据即可。

```
    years = re.search(pattern=r"\d{4}", string=OtherInfo).group()
    something = re.search(pattern=r"\d{4}.(\/).+(\/).+", string=OtherInfo)
    thing = re.compile(pattern=r"\d{4}.(\/).")
    if something is None:
        # 异常处理
        location = "无信息"
    else:
        location = thing.sub( repl: "", something.group())
    country = re.compile(pattern=r".(\/).+")
    nation = country.sub( repl: "", string=location)
    kind = re.compile(pattern=r".+(\/).")
    types = kind.sub( repl: "", string=location)

    datas.append({
        "rank": rank,
        "title": title,
        "years": years,
        "location": nation,
        "type": types,
        # "rating_star": rating_star.replace("rating", "").replace("-t", ""),
        # 获得的初始内容为rating5-t，通过replace前后内容能够得到我们想要的结果
        "rating_num": rating_num,
        "comments": comments.replace("人评价", "")
    })
return datas
```

这里需要注意的是，由于国家，剧情类型以及上映年份均在同一个 p 标签之间，则我们需要通过正则匹配来对于这个标签内容进行清洗，提取出我们想要的内容，然后就是组建 data 列表，这样便于之后我们将数据导出为 excel 文件，方便后续制图。

同时，这里在写程序时出现过一个小问题，在本图的第四行开始是一个简单的异常处理，原因是有几部国产动画片可能由于年代久远的原因，在豆瓣上并没有国家信息，在一开始没有意识到这个问题时，一直报错，显示 NoneType，在查阅相关报错信息发现了这个问题并该改正了。

```
            "location": nation,
            "type": types,
            # "rating_star": rating_star.replace("rating", "").
            # 获得的初始内容为rating5-t，通过replace前后内容能够得到我们
            "rating_num": rating_num,
            "comments": comments.replace("人评价", "")
        })
    return datas


all_datas = []
for html in htmls:
    all_datas.extend(parse_single_html(html))

df = pd.DataFrame(all_datas)
# 通过DataFrame来构造数据列表用于形成Excel文件
df.to_excel("./豆瓣电影Top250.xlsx")
# pprint.pprint(parse_single_html(htmls[0]))
```

最后一部分比较简单，就是调用函数，然后获得数据，用 pandas 自带的库将数据存入 excel 中即可。

以下是提取过后的数据列表（只截取了前 50 的部分）：

rank	title	years	location	type	rating_num	comments
1	肖申克的救赎	1994	美国	犯罪 剧情	9.7	2972691
2	霸王别姬	1993	中国大陆 中国香港	剧情 爱情 同性	9.6	2196695
3	阿甘正传	1994	美国	剧情 爱情	9.5	2215481
4	泰坦尼克号	1997	美国 墨西哥	剧情 爱情 灾难	9.5	2251779
5	这个杀手不太冷	1994	法国 美国	剧情 动作 犯罪	9.4	2346740
6	千与千寻	2001	日本	剧情 动画 奇幻	9.4	2298741
7	美丽人生	1997	意大利	剧情 喜剧 爱情 战争	9.5	1357634
8	星际穿越	2014	美国 英国 加拿大	剧情 科幻 冒险	9.4	1908859
9	盗梦空间	2010	美国 英国	剧情 科幻 悬疑 冒险	9.4	2118655
10	辛德勒的名单	1993	美国	剧情 历史 战争	9.5	1146913
11	楚门的世界	1998	美国	剧情 科幻	9.4	1768834
12	忠犬八公的故事	2009	美国 英国	剧情	9.4	1427033
13	海上钢琴师	1998	意大利	剧情 音乐	9.3	1716027
14	三傻大闹宝莱坞	2009	印度	剧情 喜剧 爱情 歌舞	9.2	1901641
15	放牛班的春天	2004	法国 瑞士 德国	剧情 音乐	9.3	1345371
16	机器人总动员	2008	美国	科幻 动画 冒险	9.3	1349249
17	疯狂动物城	2016	美国	喜剧 动画 冒险	9.2	2002005
18	无间道	2002	中国香港	剧情 犯罪 惊悚	9.3	1404407
19	控方证人	1957	美国	剧情 犯罪 悬疑	9.6	590869
20	大话西游之大圣娶亲	1995	中国香港 中国大陆	喜剧 爱情 奇幻 古装	9.2	1569112
21	熔炉	2011	韩国	剧情	9.4	953610
22	教父	1972	美国	剧情 犯罪	9.3	995006
23	触不可及	2011	法国	剧情 喜剧	9.3	1152328
24	当幸福来敲门	2006	美国	剧情 传记 家庭	9.2	1555008
25	寻梦环游记	2017	美国	喜剧 动画 奇幻 音乐	9.1	1738305
26	末代皇帝	1987	英国 意大利 中国大陆 法国	剧情 传记 历史	9.3	914839
27	龙猫	1988	日本	动画 奇幻 冒险	9.2	1296010
28	怦然心动	2010	美国	剧情 喜剧 爱情	9.1	1877803
29	活着	1994	中国大陆 中国香港	剧情 历史 家庭	9.3	872021
30	哈利·波特与魔法石	2001	美国 英国	奇幻 冒险	9.2	1233055
31	蝙蝠侠：黑暗骑士	2008	美国 英国	剧情 动作 科幻 犯罪 惊悚	9.2	1089943
32	指环王3：王者无敌	2003	美国 新西兰	剧情 动作 奇幻 冒险	9.3	825316
33	我不是药神	2018	中国大陆	剧情 喜剧	9.0	2155532
34	乱世佳人	1939	美国	剧情 历史 爱情 战争	9.3	710124
35	飞屋环游记	2009	美国	剧情 喜剧 动画 冒险	9.1	1361695
36	素媛	2013	韩国	剧情	9.3	705760
37	十二怒汉	1957	美国	剧情	9.4	508082
38	哈尔的移动城堡	2004	日本	动画 奇幻 冒险	9.1	1056227
39	让子弹飞	2010	中国大陆 中国香港	剧情 喜剧 动作 西部	9.0	1743719
40	何以为家	2018	黎巴嫩 美国 法国 塞浦路斯 卡塔尔 英国	剧情	9.1	1070746
41	摔跤吧！爸爸	2016	印度	剧情 传记 运动 家庭	9.0	1604279
42	猫鼠游戏	2002	美国 加拿大	传记 犯罪 剧情	9.1	1063019
43	天空之城	1986	日本	动画 奇幻 冒险	9.2	902770
44	鬼子来了	2000	中国大陆	剧情 喜剧	9.3	643894
45	海蒂和爷爷	2015	德国 瑞士	剧情 冒险 家庭	9.3	647561
46	少年派的奇幻漂流	2012	美国 中国台湾 英国 加拿大	剧情 奇幻 冒险	9.1	1378626
47	钢琴家	2002	英国 法国 波兰 德国	剧情 传记 战争 音乐	9.3	662915
48	大话西游之月光宝盒	1995	中国香港 中国大陆	喜剧 爱情 奇幻 古装	9.0	1249483
49	指环王2：双塔奇兵	2002	美国 新西兰	剧情 动作 奇幻 冒险	9.2	774532
50	闻香识女人	1992	美国	剧情	9.1	914443

其次，由于数据十分多，同时包含文本类数据，为了对这些数据进行处理，以便能够尽量利用到这些数据，我对于数据做了一些重新的处理。

1） 从刚刚的 excel 表重新提取数据：

```python
import re

import pandas as pd

data = pd.read_excel(r'D:\Dase1\final\豆瓣电影Top250.xlsx')
data = data.iloc[:, :]
ranks = data['rank'].tolist()
years = data['years'].tolist()
locations = data['location'].tolist()
styles = data['type'].tolist()
nums = data['rating_num'].tolist()
comments = data['comments'].tolist()
USAs = []
UKs = []
Japans = []
Koreans = []
Frances = []
CNs = []
Other_countries = []
# 动画、剧情、爱情、动作、科幻、喜剧、冒险、历史
cartoons = []
features = []
loves = []
moves = []
sfs = []
comics = []
advs = []
histories = []
```

这一部分首先从刚刚保存好的数据表中提取了数据，然后按列提取成多个列表。对于后续的处理内容做了一些初始化。比如，对于国家，我进行了简单分类，分为美，英，中，日，韩，法，其他；根据类型，分为了剧情，爱情，冒险，动作，动画，历史，科幻，喜剧。根据这些分类进行了简单的初始化

接下来就是对各个分类中的种类数量去计数，同样用到了正则匹配，因为国家和电影类型的字段值都存在多个并存的情况，故需要通过正则匹配来检索并计数。代码如下：

```python
for location in locations:
    tag = 1
    if re.search(pattern="美国", string=location) is not None:
        tag = 0
        USAs.append(location)
    if re.search(pattern="英国", string=location) is not None:
        tag = 0
        UKs.append(location)
    if re.search(pattern="日本", string=location) is not None:
        tag = 0
        Japans.append(location)
    if re.search(pattern="韩国", string=location) is not None:
        tag = 0
        Koreans.append(location)
    if re.search(pattern="法国", string=location) is not None:
        tag = 0
        Frances.append(location)
    if re.search(pattern="中国", string=location) is not None:
        tag = 0
        CNs.append(location)
    if tag == 1:
        Other_countries.append(location)

countries = []
countries.append({
    "USA": len(USAs),
    "UK": len(UKs),
    "CN": len(CNs),
```

```
})

for style in styles:
    tag = 1
    if re.search(pattern="剧情", string=style) is not None:
        tag = 0
        features.append(style)
    if re.search(pattern="爱情", string=style) is not None:
        tag = 0
        loves.append(style)
    if re.search(pattern="科幻", string=style) is not None:
        tag = 0
        sfs.append(style)
    if re.search(pattern="动画", string=style) is not None:
        tag = 0
        cartoons.append(style)
    if re.search(pattern="动作", string=style) is not None:
        tag = 0
        moves.append(style)
    if re.search(pattern="历史", string=style) is not None:
        tag = 0
        histories.append(style)
    if re.search(pattern="冒险", string=style) is not None:
        tag = 0
        advs.append(style)
    if re.search(pattern="喜剧", string=style) is not None:
        tag = 0
        comics.append(style)
```

最后也是简单的将数据组成，然后导入到 excel 中：

```
Kinds = []
Kinds.append({
    "爱情": len(loves),
    "剧情": len(features),
    "动作": len(moves),
    "科幻": len(sfs),
    "历史": len(histories),
    "动画": len(cartoons),
    "冒险": len(advs),
    "喜剧": len(comics)
})

df = pd.DataFrame(countries)
# 通过DataFrame来构造数据列表用于形成Excel文件
df.to_excel("./豆瓣电影Top250国家分布.xlsx")

df = pd.DataFrame(Kinds)
# 通过DataFrame来构造数据列表用于形成Excel文件
df.to_excel("./豆瓣电影Top250剧情分布.xlsx")
```

以下是获得的两个数据表：

爱情	剧情	动作	科幻	历史	动画	冒险	喜剧
57	184	31	23	7	36	50	52

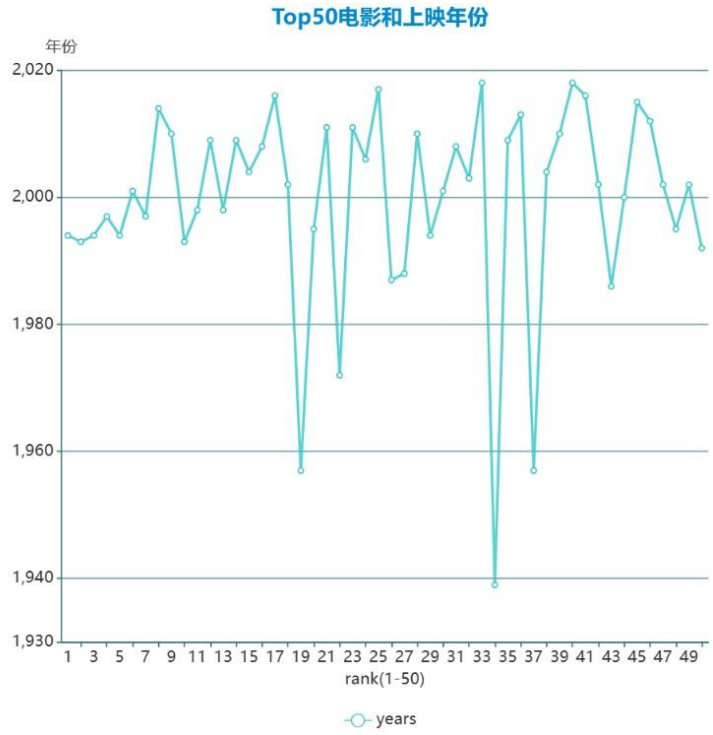
豆瓣电影 Top250 剧情类型分布

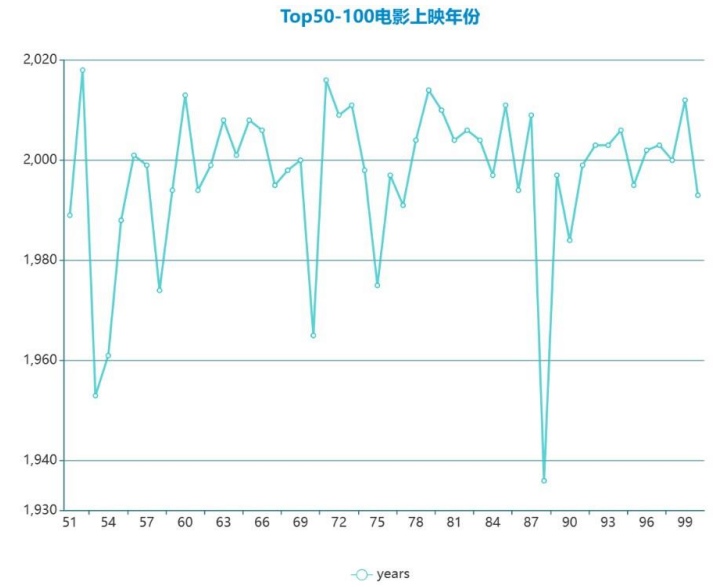
USA	UK	CN	Korean	France	Japan	Else
142	39	46	11	18	35	14

豆瓣电影 Top250 国家分布

三、数据可视化部分

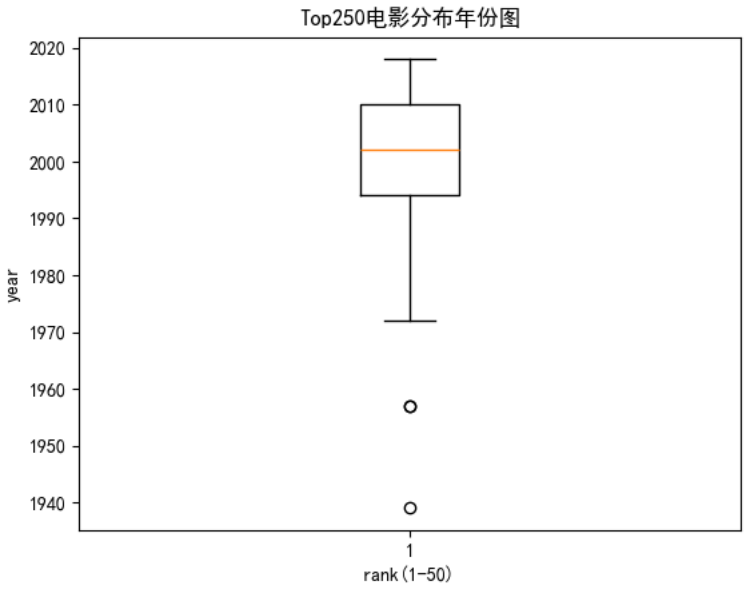
首先我们看看 Top50 电影上映年份和 Top50-100 电影上映年份的折线图；



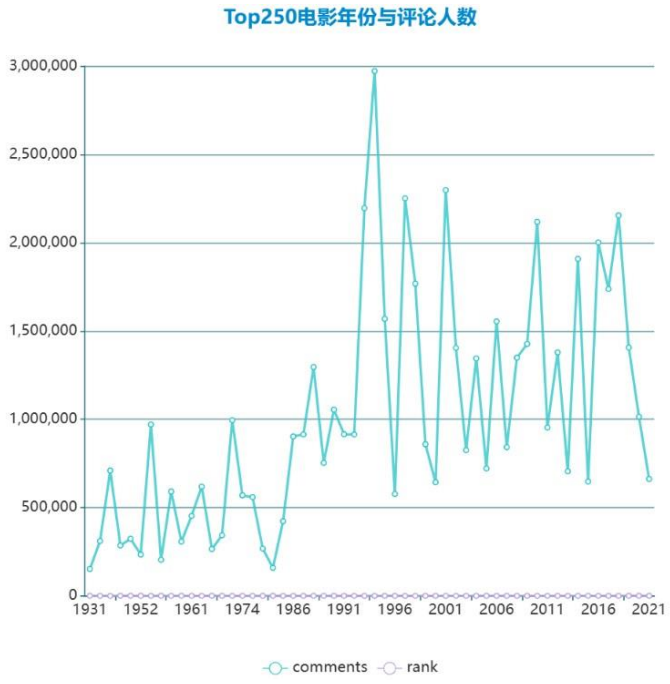


从这两张图可以看出总体上排名靠前的电影都是在 2000 年-2011 年之间上映的，接下来占比较大的部分便是 1990 年-2000 年的部分，那么是否意味着随着时间的推进，中国影民的口味表明近年来好电影越来越少了呢，尤其是近几年的电影没有一部上榜这个评分。

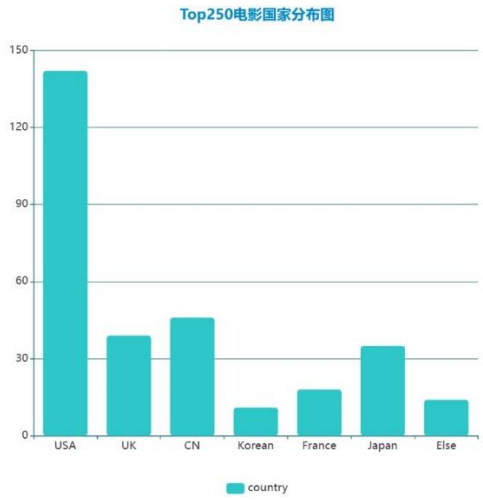
同时。根据下面的箱型图也能发现 Top250 电影的分布主要确实实在 1990-2000 年代，特别是 2000 年附近涌现了许多优质电影。

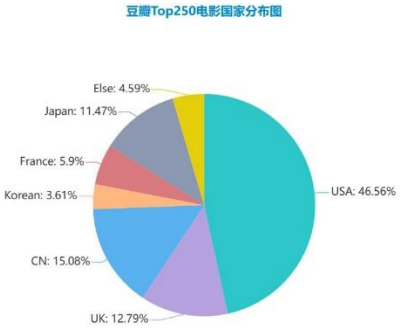


接下来我们再看看电影的热度指标，也就是评论数能不能反映一些什么：

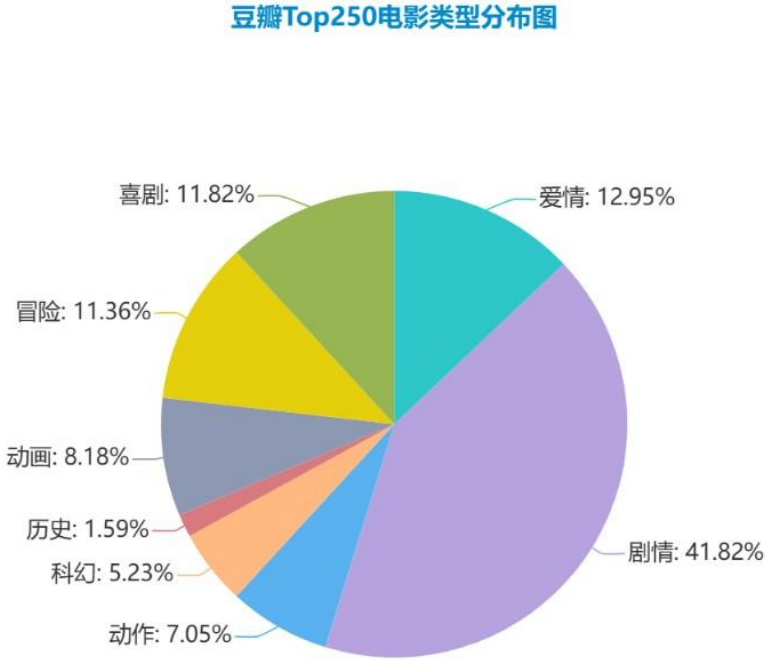


根据上图，我们发现，虽然人口在逐年上升，新生代群体越来越多，但按照上映年份排序的评论人数折线图表明电影人气最高的几部似乎都出现在 90 年代到 00 年代初，比如排名第一的《肖生克的救赎》诞生于 1994 年。这一部分表明影民对于高质量经典电影的认可，这些电影对于后来的群众依然保持了无比的吸引力；另一方面，近十年的电影评论人数似乎不低，但鉴于影民数量的大量提升，这样的评论人数大致也能反映近几年世界电影的发展渐缓，可能一方面与电影受众的审美要求提升有关。也反映了如今电影界能够真正流芳的电影日趋减少。





以上两张分别是 Top250 电影的国家分布的柱状和饼状图，我们可以发现欧美电影尤其是美国，还是占据了 Top250 电影的大部分，这一方面与欧美电影业发展早，电影工业化成熟的缘故；另一方面，日本电影在其中的占比与国产电影相差无几，也侧面反映了日本电影对于中国影民也具有不错的吸引力，例如宫崎骏的动画电影就是很好的例子。



最后是 Top250 电影类型的分布图。从这张图我们能看到剧情片的吸引力十分大，这也有可能是大部分电影的类型标签本身都有剧情。去除这个可能得干扰项后，我们可以看到：

根据这张图我们能看到爱情，喜剧和冒险电影更加吸引中国影民的口味偏好，再加上动画，说明国人的电影偏好更偏向于合家欢的类型。

四、总结

根据以上的图表分析，我们不难得出优质电影在近十年的涌现率已经慢慢不如再往前的两个十年了，一方面可能是大众电影的审美等级提升，另一方面也与近几年电影界良心制作越来越少，消费主义等影响了优质电影的发掘，叫好又叫座成为了一个难以兼得的问题。

此外，在完成此作品的过程中，如果能够对于比如国家的分类中，将国家与排行数据结合这样的尝试，应该能得到更多新的发现，在后续可以继续尝试改进。