基于Python的豆瓣影评数据爬取与分析

一、数据爬取

页面分析

本次要爬的网站是豆瓣电影 Top250，这是网友评选的 250 部最佳影片。网址是https://movie.douban.com/top250?start=50&filter=

首先，我们要爬的信息有，电影名，电影别名，导演，影片制作国家，所属类型，评分，评价数量，一句话评价。如下图，我们要的信息在图中都能看到。

通过对网页源码的分析，可以发现要爬的每个页面信息都在 ol 属性，而每部电影的基本信息都在 class 为 info 的 div 属性中。这样我们就可以爬取每个页面中所有 class 为 info 的 div 属性，获取文本，然后进行提取信息。之后要爬取其他页面，发现了页面的链接是有规律的。
然后对下几页的url分析，很容易发现只是start值以每次25递增，所以我们只要改动start的值，就能获取到需要抓取的所有页面的url了

爬取数据

获取html页面

导入requests库，向服务器发送get请求，返回网页源代码

```
html = requests.request('get', url=pageUrl, headers=headers)
# 设置网页编码格式
html.encoding = 'utf8'
```

解析数据

在这里我们导入lxml库,通过xpath方法可以比较准确的抓取到我们需要的数据

```
text = html.text
tree = etree.HTML(text)

items = tree.xpath('//ol/li/div/div[@class="info"]')

for item in items:
    #电影名
```

```
    name = item.xpath("./div[@class= 'hd']/a/span[1]/text()")[0]
    #电影别名
    otherName = item.xpath("./div[@class='hd']/a/span[@class='other']/text()")
    #过滤后的电影别名
    clean_OtherName = otherName[0].split('/',1)[1].replace('  ',"").lstrip()
    #导演各种信息
    bd_info = item.xpath("./div[@class='bd']/p[1]/text()")#返回一个列表
    #导演
    director = bd_info[0].replace("\n", "").split('   ')
    clean_director = director[0].split(':')[1].strip()

    info_1 = bd_info[1].strip().replace("\n", "").split(' / ')
    #年份
    year = info_1[0].strip()
    #国家
    country = info_1[1].strip()
    #类型
    type = info_1[2].strip()
    #评分
    rating = item.xpath("./div[@class='bd']/div[@class='star']/span[2]/text()")
[0]
    #评分人数
    evaluation_num =
item.xpath("./div[@class='bd']/div[@class='star']/span[4]/text()")
    clean_evaluation_num =re.findall(r'\d+',evaluation_num[0])[0]
    #一句话简介
    quote_tag = item.xpath("./div[@class='bd']/p/span[@class='inq']")
    # 注意可能可能没有quote简评
    if len(quote_tag) is not 0:
        quote = quote_tag[0].text
    else:
        quote=""

    list =
[i,name,clean_OtherName,clean_director,year,country,type,rating,clean_evaluation
_num,quote]
    i = i+1
    allDataList.append(list)
```

数据保存

在这里我们通过csv库，将爬取的数据保存到movies.csv文件中

```
 #将数据写入csv文件
def writeToCsv(self,allDataList):

    path = r'./data/'
    if not os.path.exists(path):
        os.mkdir(path)

    with open(path+"movies.csv","w",encoding='utf-8-sig',newline="") as csvfile:
        writer = csv.writer(csvfile)
        #列名
        writer.writerow(['Top250','MovieName', 'OtherName', 'Director',
                         'Year', 'Country', 'Type', 'Rating', 'EvaluationNum',
                         'quote'])
        writer.writerows(allDataList)
```

二、数据可视化

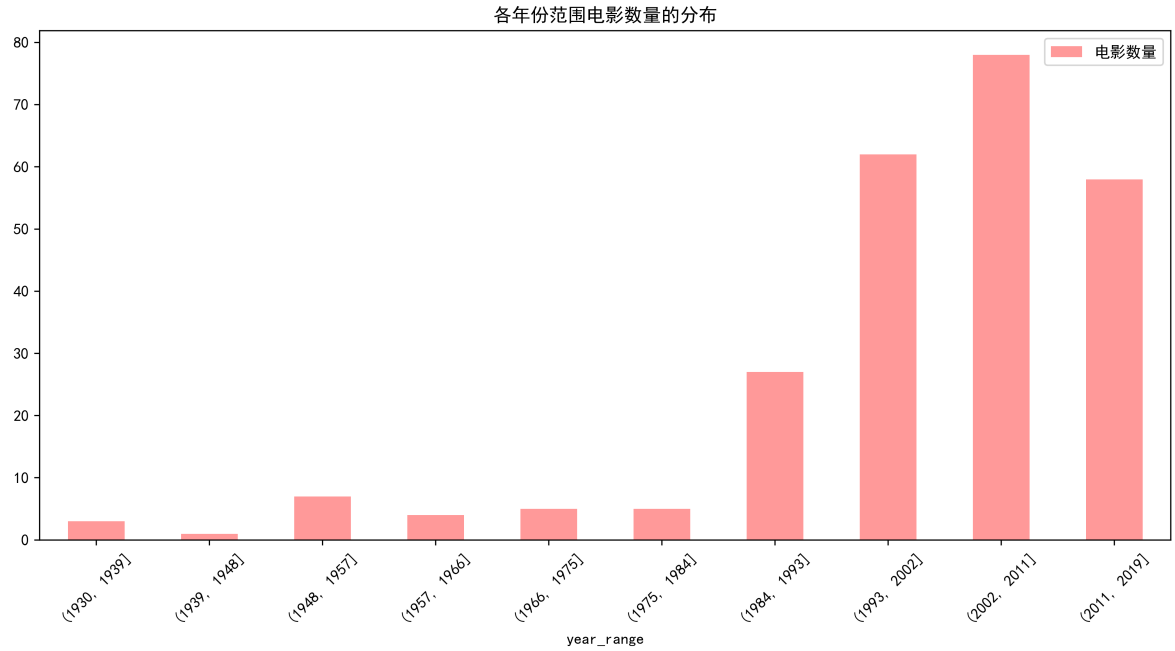
电影上映年份分析

因为榜单中电影上映年份跨度比较大，因此我们利用pandas模块对数据进行分组后。统计数据可视化输出。

```
# 按年份进行分组
def year_group(year):
    # 判断，有两部电影的年份包含制作国家
    if len(year) > 4:
        year = year[0:4]
    year = int(year)

    if (year > 1930) and (year <= 1939):
        return '(1930, 1939]'
    elif (year > 1939) and (year <= 1948):
        return '(1939, 1948]'
    elif (year > 1948) and (year <= 1957):
        return '(1948, 1957]'
    elif (year > 1957) and (year <= 1966):
        return '(1957, 1966]'
    elif (year > 1966) and (year <= 1975):
        return '(1966, 1975]'
    elif (year > 1975) and (year <= 1984):
        return '(1975, 1984]'
    elif (year > 1984) and (year <= 1993):
        return '(1984, 1993]'
    elif (year > 1993) and (year <= 2002):
        return '(1993, 2002]'
    elif (year > 2002) and (year <= 2011):
        return '(2002, 2011]'
    elif (year > 2011) and (year <= 2019):
        return '(2011, 2019]'

# 对电影的上映年份进行分析统计
def analyze_year():
    # pandas显示所有列或所有行
    pd.set_option('display.max_rows', None)
    df['year_range'] = df['Year'].apply(year_group)
    year = df.groupby('year_range')['MovieName'].count()
    index= year.index
    values = year.values

    ts = pd.Series(values, index=index)
    ts.plot(
        kind='bar',
        title = '各年份范围电影数量的分布',
        rot = 45,
        color = 'red',
        alpha = 0.4,
        figsize=(13, 6)
    )
    plt.legend(['电影数量'])
    # 输出图表
    plt.savefig('./charts/各年份范围电影数量的分布.png',
                dpi=400,
```

```
                bbox_inches = 'tight')
```

绘制结果如下：

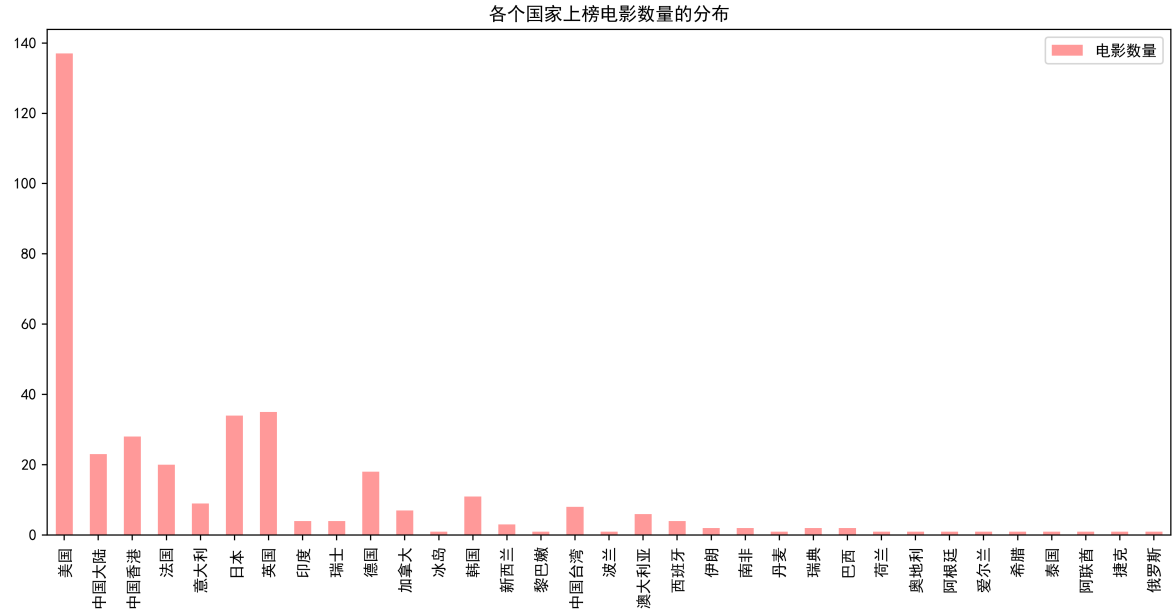


由数据可以看出，2002-2011年上映的电影上榜数最多

电影制作国家分析

```
# 对电影的制作国家进行分析统计
def analyze_country():
    #lambda表达式进行以空格分开，并返回列表
    countryList = reduce(lambda x, y: x + y, list(df['Country'].apply(lambda x:
x.split(' '))))
    #进行分类统计
    countryDic = Counter(countryList)
    #将字典转为DataFrame
    country_dataframe = pd.DataFrame.from_dict(countryDic, orient='index')

    country_dataframe.plot(
        kind='bar',
        title = '各个国家上榜电影数量的分布',
        rot = 90,
        color = 'red',
        alpha = 0.4,
        figsize=(13, 6)
    )
    plt.legend(['电影数量'])
    # 输出图表
    plt.savefig('./charts/各个国家上榜电影数量的分布.png',
                dpi=400,
                bbox_inches = 'tight')
```

绘制结果如下：



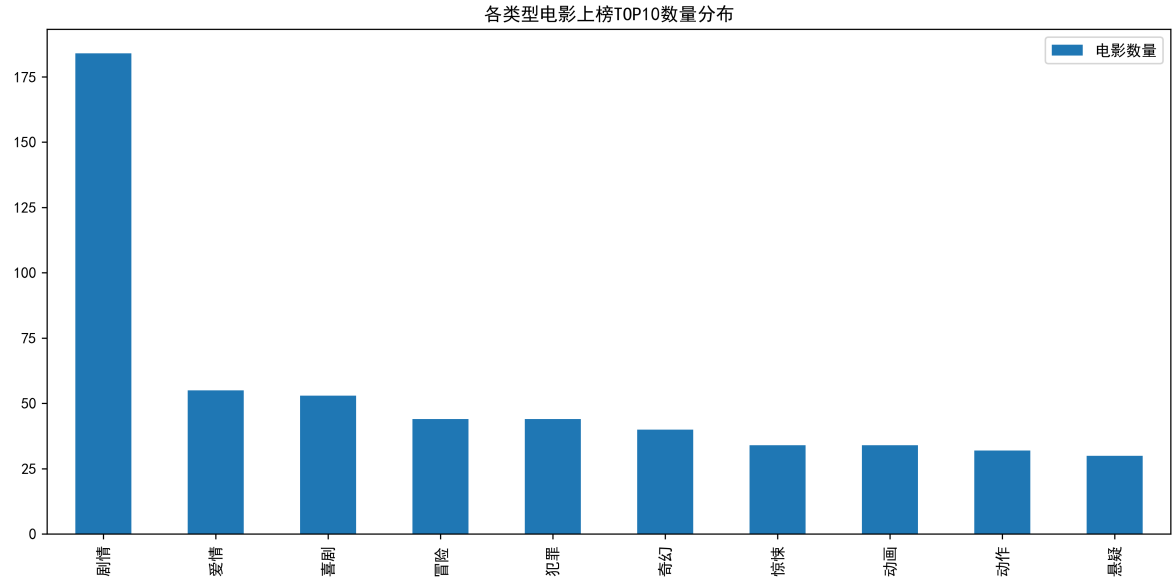
由数据图可以看出，美国稳居榜首，中国电影上榜数也比较多。

电影类型分析

因为有些电影属于多种类型。比如排在第一位的《肖申克的救赎》既属于犯罪片也属于剧情片，而排在第二位的《霸王别姬》既属于剧情片又属于爱情片与同性片。对于这种情况，采用split方法对每一项国家或地区数据进行切割，并将空值NaN替换为"0"，先按列计数，再按行汇总，由此统计数量。

```
# 对电影不同类型进行分析统计
def analyze_type():
    # 电影类型统计
    all_type = df['Type'].str.split(' ').apply(pd.Series)
    type_text = all_type.to_string(header=False, index=False)
    all_type = all_type.apply(pd.value_counts).fillna('0')
    all_type.columns = ['type1', 'type2', 'type3', 'type4', 'type5']
    all_type['type1'] = all_type['type1'].astype(int)
    all_type['type2'] = all_type['type2'].astype(int)
    all_type['type3'] = all_type['type3'].astype(int)
    all_type['type4'] = all_type['type4'].astype(int)
    all_type['type5'] = all_type['type5'].astype(int)
    all_type['all_counts'] = all_type['type1'] + all_type['type2'] \
                             + all_type['type3'] + all_type['type4'] +
all_type['type5']

    all_type = all_type.sort_values(['all_counts'], ascending=False)
    # 取电影类型前10做分析
    movie_type = pd.DataFrame({'数量': all_type['all_counts']})[:10]
#     print(movie_type[:10])
    movie_type.plot(
                    kind='bar',
                    figsize=(13, 6),
                    title = '各类型电影上榜TOP10数量分布')
    plt.legend(['电影数量'])
    # 输出图表
    plt.savefig('./charts/各类型电影上榜TOP10数量分布.png',
                dpi=400,
                bbox_inches = 'tight')
```

绘制结果如下：

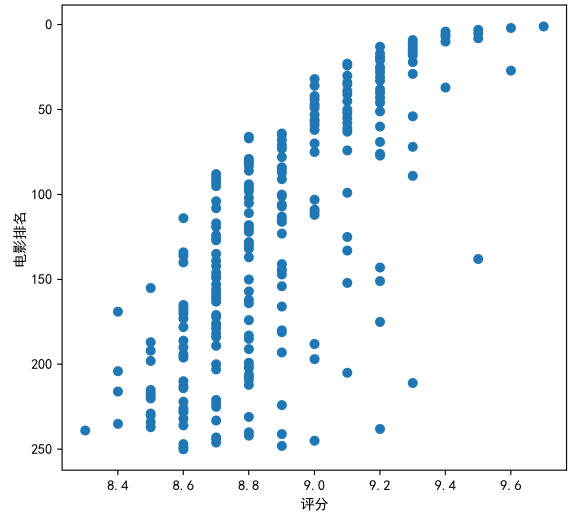


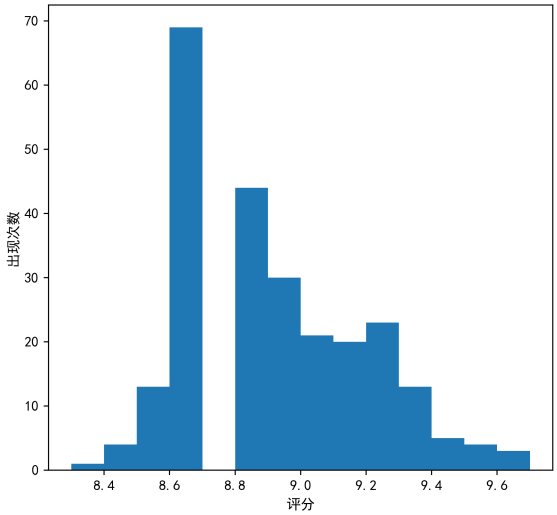
由数据图可以看出，剧情片上榜数量远远超过其他类型，而情色片，运动片等类型上榜较少

评分与排名关系分析

```
# 评分与排名关系分析
def analyze_rating():
#     评分与排名的关系散点图
    plt.figure(figsize=(14, 6))
    plt.subplot(1, 2, 1)
    plt.scatter(df['Rating'], df['Top250'])
    plt.xlabel('评分')
    plt.ylabel('电影排名')
    plt.gca().invert_yaxis()
    # 评分数量直方图
    plt.subplot(1, 2, 2)
    plt.hist(df['Rating'], bins=14)
    plt.xlabel('评分')
    plt.ylabel('出现次数')
    plt.savefig('./charts/评分与排名关系分析.png',
                dpi=400,
                bbox_inches = 'tight')
```

绘制结果如下：



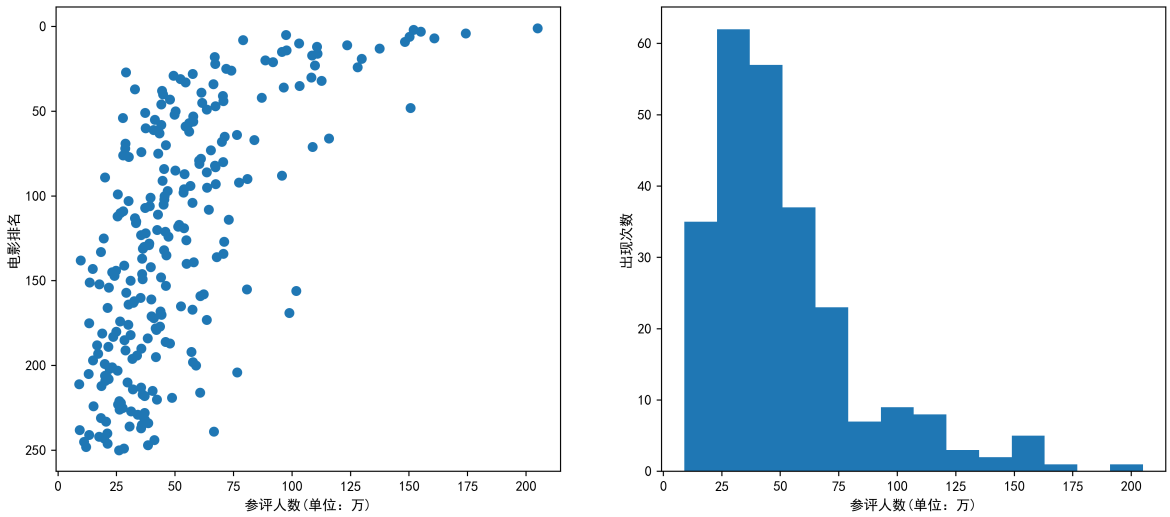


由上图可以看出，评分与出现频率基本呈正态分布

评分与评价人数关系分析

```
# 评分与评价人数关系分析
def analyze_evaluationNum():
    #      评分与评价人数的关系散点图
    plt.figure(figsize=(14, 6))
    plt.subplot(1, 2, 1)
    l1 = list(i / 10000 for i in df['EvaluationNum'].values)
    plt.scatter(l1, df['Top250'])
    plt.xlabel('参评人数(单位：万)')
    plt.ylabel('电影排名')
    plt.gca().invert_yaxis()
    # 评分数量直方图
    plt.subplot(1, 2, 2)
    plt.hist(l1, bins=14)
    plt.xlabel('参评人数(单位：万)')
    plt.ylabel('出现次数')
    plt.savefig('./charts/评分与评价人数分析.png',
                dpi=400,
                bbox_inches = 'tight')
```

绘制结果如下：



由数据图可以看出，电影排名与参评人数基本呈正相关趋势。

主题分析

```
# 对热门短评进行主题分析，生成词云
def comment_wordcloud():
        text = ''
        for i in df['quote'].values:
            text += str(i)

        counts = {}
        words = jieba.lcut(text)
        for word in words:
            if len(word) == 1:
                continue
            counts[word] = counts.get(word, 0) + 1
        # print(counts)
        wcloud = wordcloud.WordCloud(
            font_path='./common/SimSun.ttf',
            background_color= 'white',width=1000,
```

```
            max_words= 50,
            height= 860, margin= 1
        ).fit_words(counts)
        wcloud.to_file('短评词云.png')
```

结果如下：

三、总结

综上分析，可以看出，中国观众对于电影的类型更认可剧情片和爱情片；对于电影的生产国家，更认可具有强大电影工业体系的美国；对于电影的上映年份与上榜数量关系可以看出，随着社会的发展，人们的目光也渐渐由满足与物质需求转向精神需求。而对于该榜单中的电影而言，好电影会吸引更多的人来评价，而评价人数多的电影，评价往往也不差。