[热门电影短评数据爬取与分析](https://edu.tipdm.org/my/course/2840)

中国电影最近几年突飞猛进，越来越多的人走进电影院了，各个大盘影片轻轻松松就能突破几十亿票房，但是随着电影消费的增加，大家对电影质 量和制作水平的要求也提高了很多，想要继续斩获高票房，就得把握好消费者的喜好，制作出符合市场期待的电影。怎样了解消费者偏好是电影出 品方关注的事情，而豆瓣作为国内主流电影评价网站，拥有关于电影的评论、排行、评分等信息，是消费者喜欢的评价电影主流方式，也是电影出品方喜欢的了解用户喜好的途径。所以掌握豆瓣电影短评的获取方法有助于了解电影市场，帮助出品方规划未来电影拍摄计划，对出品方投资等有非常重要的意义。

**学习目标**

1. 获取《复仇者联盟4》的短评数据。
2. 将数据做预处理，清洗数据，得到自己想要的数据，如用户长居城市，注册时间等。对短评进行词云统计，绘制词云图
3. 对发表短评的时间进行分析，探讨发表时间变化情况，分析不同城市的评分状况。
4. 用户会龄的评分影响。

# 爬取《复仇者联盟4》的短评数据

**任务描述**

本次任务通过python的requests库对豆瓣中的数据进行爬取，然后使用beautifulsoup库对数据信息进行解析。

**任务分析**

由于豆瓣是比较经典的电影论坛，因此我们选择豆瓣作为我们爬取数据的对象，但是豆瓣官方有一些反扒机制。在爬取豆瓣短评时候，我们在没有登陆的状态下，至多能够爬取200条数据，而且我们没法找到登录的接口，因此，我们没法爬取到太多的数据，而且即使在登录状态下，我们至多也只能爬取到500条数据。

首先我们分析了豆瓣的短评页面，由于它是分页查询供用户查看的，因此我们使用for循环不断发送请求，一页页地爬取数据，这里我们需要爬取10次，每次20条，一共两百条。

我们使用reuests.get方法发送get请求，获取到页面信息，然后分析页面结构，然后通过beautifulsoup解析页面，从而获取希望得到的数据。这里也会有一些坑，由于某些数据是有可能没有的，比如星级。当某些数据不存在时，我们不能直接获取整个页面的数据，我们要对每一条数据进行获取，如果获取不到该用户的某一数据，则我们将放弃该用户的短评，就是说，如果无法获取该用户某数据，则将捕获到异常，则跳过该用户。

每个用户还有其用户详细页地接口，所以我们还需要获取该用户的详细页接口再发送请求，获取页面信息，然后解析，得到用户详细信息。也就是说每个页面20个用户短评信息，我们还用对这20个用户爬取其详细信息。

这里我们一共获取到的数据有这些：comments，usernames，starts，times，hasuses，userhrefs，citys，intodoubantimes

1. 分析豆瓣短评接口，从而不断发送请求爬取分页数据。
2. 对爬取到的页面分析，使用beatufulsoup进行解析。
3. 解析完成后，获取想要获取的数据，用pandas构建数据框，并保存起来。

## 分析豆瓣短评接口，从而不断发送请求爬取分页数据

豆瓣短评接口是： 'https://movie.douban.com/subject/1866479/comments?start={}&limit=20&sort=new\_score&status=P'，其中start表示起始页，limit即为分页查询，每页20条短评数据。因此我们使用for循环不断爬取数据。

## 对爬取到的页面分析，使用beatufulsoup进行解析

Beatufulsoup有很好的解析性，主要它可以像前端一样对获取到的页面构建dom树，然后通过css选择器获取数据内容。具体代码如下：

|  |
| --- |
| for com in soup.select('#comments div.comment'):  try:  comment = com.select('span.short')[0].text  username = com.select('h3 > span.comment-info > a')[0].text  start = com.select('h3 > span.comment-info > span.rating')[0]['class']  time1 = com.select('h3 > span.comment-info > span.comment-time')[0]['title']  hasuse = com.select('h3 > span.comment-vote > span')[0].text  userhref = com.select('h3 > span.comment-info > a')[0]['href']  except:  continue  else:  comments.append(comment)  usernames.append(username)  starts.append(start)  times.append(time1)  hasuses.append(hasuse)  userhrefs.append(userhref) |

## 解析完成后，获取想要获取的数据，用pandas构建数据框

获取我们希望获取到的数据，当数据不存在时候可以用NAN来表示，这里我们获取到的数据信息有：comments表示短评信息，usernames表示用户名，starts表示评分，times表示发表评论时间，hasuses表示点赞的个数，userhrefs表示用户详细页链接，citys表示用户所在地址，intodoubantimes表示注册时间。表1 - 1为部分爬取到的数据。

表 1‑1 短评数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **comments** | **usernames** | **starts** | **times** | **hasuses** | **citys** | **intodoubantimes** |
| 如果你不喜欢这部电影，说明他不是为你准备的，故事的终章是为读过故事的人准备的 | [棠枫海](https://www.douban.com/people/146803809/) | 501 | 2019-04-24 | 36346 | [山西太原](https://www.douban.com/location/taiyuan/) | 2016-06-08 |
| 我是一个90后，我曾经很羡慕“上一代人”：40年前的观众，他们的影院里有星战正传三部曲的落幕；20年前的观众，能在影院里看到指环王系列的终章。影迷们的悲欢并不相通，但现在我们也有了共同的记忆：漫威电影宇宙。谢谢你，《复仇者联盟》。 | Tel | 50 | 2019-04-24 | 20105 | [北京](https://beijing.douban.com/) | 2017-07-21 |
| 钢铁侠成为了美队，美队活成了钢铁侠。 | [Rafe](https://www.douban.com/people/Rafe0323/) | 50 | 2019-04-24 | 14014 | [重庆](https://www.douban.com/location/chongqing/) | 2015-02-27 |
| 谁能想到是一只老鼠拯救了地球呢？ | [KarSa](https://www.douban.com/people/karsa/) | 50 | 2019-04-24 | 12503 | 上海 | 2011-3-10 |
| 献给我人生中最美好的十一年。谢谢你，漫威，谢谢你让我的青春有了一个最完美的结局 | 。 | 50 | 2019-04-24 | 13417 | Nan | 2018-10-21 |

# 对数据进行预处理，对短评进行词云统计，绘制词云图

**任务描述**

我们希望对短评进行词云统计，那么在此之前首先就是要将我们获取到的数据处理干净，然后再进行分词，去停用词，词频统计，最后再绘制词云图。

**任务分析**

对短评数据操作可以分为以下2个步骤。

1. 对数据进行预处理。
2. 对短评进行分词，去停用词，词频统计，绘制词云图等。

## 对数据进行预处理

首先我们再存入csv的时候是分页存入的，也就是没有表头存入，所以我们读取到数据后，先要给数据加上表头，即'comments','username','starts','times','hasuses','citys','intodoubantimes'。

其次还需要将时间格式化成datetime,这样方便我们后面对时间的统计。

Start列获取下来是一窜字符串，如：['allstar40', 'rating']，因此我们需要将里面的‘40’截取下来，是我们需要的。

还有城市citys列里面有些国外的城市是英文的，分大小写，我们需要将其全变成小写的。具体代码如下：

|  |
| --- |
| #加上列名  data.columns = ['comments','username','starts','times','hasuses','citys','intodoubantimes']  data.index = range(len(data))  #处理start列变为数字  data['starts'] = data['starts'].apply(lambda x: int(re.search('\d+',str(x)).group())).fillna(0)  #将时间转换成datatime  data['times'] = pd.to\_datetime(data['times'])  #处理citys  data['citys'] = data['citys'].fillna('')  data['citys'] = data['citys'].str.lower().apply(lambda x: str(x).split(',')[0])  #处理加入豆瓣的时间  def formatTimeToDouban(x):  res = re.search('\d+-\d+-\d+',str(x))  if res:  return res.group()  return np.nan  data['intodoubantimes'] = data['intodoubantimes'].apply(formatTimeToDouban) |

## 对短评进行分词，去停用词，词频统计，绘制词云图

对短语分词，是我们词云统计的一个重要的步骤，首先分词我们使用的是jieba库，分完词后会得到一些常见的词语，但这还不是我们想要的，有些词语并没有太大作用，因此我们还需要将一些没有的词去掉，比如单个出现的英文‘a’,’b’，或是单个字‘的’等等，而后我们还需要进行词云统计，统计完成后，就是绘制词云图了。详细代码如下：

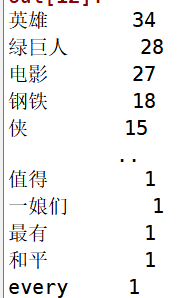
|  |
| --- |
| #评论分词并且去掉停用词操作  with open('../01.中间数据（如有）/stopword.txt', 'r') as f:  stopword1 = f.read()  with open('../01.中间数据（如有）/stoplist.txt', 'r',encoding='utf-8') as f:  stopword2 = f.read()  a = data['comments'].apply(lambda x: x.lower()).apply(lambda x: ''.join(x)).apply(jieba.lcut).apply(lambda x: [i for i in x if i not in stopword1]).apply(lambda x: [i for i in x if i not in stopword2])  a[a.apply(lambda x: len(x) < 6)] = np.nan  data['comments'] = a  #词频统计  def get\_word\_cloud(data=None,colname=None):  words = []  describe = data[colname].str[1:-1]  describe.dropna(inplace=True)  [words.extend(i) for i in describe]  words = pd.Series(words)  word\_fre = words.value\_counts()  return word\_fre  word\_fre = get\_word\_cloud(data=data, colname='comments')  #绘制词云图  mask = plt.imread('../01.中间数据（如有）/aixin.jpg')  wc = WordCloud(mask=mask, background\_color='white', font\_path=r'../01.中间数据（如有）/simhei.ttf')  wc.fit\_words(word\_fre)  plt.imshow(wc) |

首先我们分完词后，去除停用词来自数据文件夹中的stop\_word.txt以及stop\_list，因为两个问价比较多的词，所以效果还算比较好，图2.2即我们的效果图

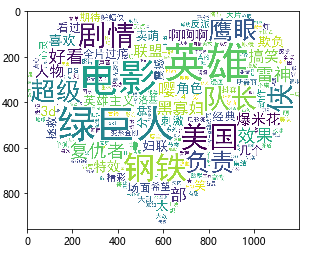


图 2‑2 分词后效果图

然后我们对这里所有的词语进行词云统计，得到效果如下：



最后就是绘制词云图了，对所有的评论绘制词云图效果如图2.2.1



**图 2‑2-1 对所有词语绘制的词云图**

当然对所有的词语绘制其词云图，我们可以很清晰地看出这是一步关于《复联4》的电影，但是没法看出那些是属于好评的，哪些是差评的，因此我们还将数据分成’好评‘，’差评‘，’一般‘三个评分等级，然后再绘制其评分对应的词云图。

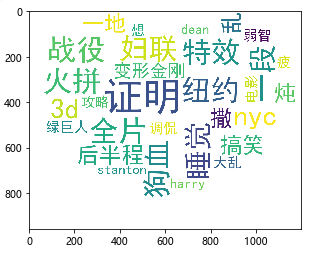
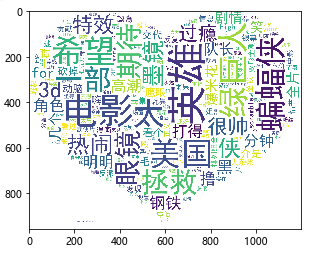
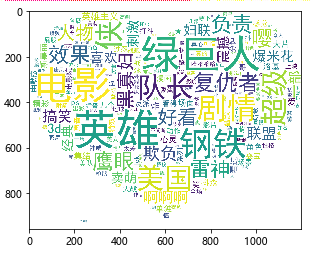
所以我们首先需要对数据框增加一个列---’leval‘，表示评分等级，我将大于等于40分的表示为好评，等于30分的表示一般，小于30分表示差评。具体代码如下：

|  |
| --- |
| def getLeval(x):  if x >= 40:  return '好评'  elif x == 30:  return '一般'  else:  return '差评'  data['leval'] = data['starts'].apply(getLeval)  def get\_word\_cloud(data=None,colname=None,tianjian=None):  words = []  describe = data[colname][data['leval'] == tianjian].str[1:-1]  describe.dropna(inplace=True)  [words.extend(i) for i in describe]  words = pd.Series(words)  word\_fre = words.value\_counts()  return word\_fre |

实现这一步后，我们再对每个评分等级绘制其对应词云图，具体代码如下：

|  |
| --- |
| #好评  word\_fre = get\_word\_cloud(data=data, colname='comments',tianjian='好评')  mask = plt.imread('../01.中间数据（如有）/aixin.jpg')  wc = WordCloud(mask=mask, background\_color='white', font\_path=r'../01.中间数据（如有）/simhei.ttf')  wc.fit\_words(word\_fre)  plt.imshow(wc)  # 一般  word\_fre = get\_word\_cloud(data=data, colname='comments',tianjian='一般')  mask = plt.imread('../01.中间数据（如有）/aixin.jpg')  wc = WordCloud(mask=mask, background\_color='white', font\_path=r'../01.中间数据（如有）/simhei.ttf')  wc.fit\_words(word\_fre)  plt.imshow(wc)  # 差评  word\_fre = get\_word\_cloud(data=data, colname='comments',tianjian='差评')  mask = plt.imread('../01.中间数据（如有）/aixin.jpg')  wc = WordCloud(mask=mask, background\_color='white', font\_path=r'../01.中间数据（如有）/simhei.ttf')  wc.fit\_words(word\_fre)  plt.imshow(wc)  #处理完成，保存  data.to\_csv('../01.中间数据（如有）/comments\_info1.csv',encoding='utf\_8\_sig') |

效果图如图2.2.2，可以看到效果其实并没那么好，因为数据量的原因，我们的数据太少了，好评的稍微比较好一点，差评的效果并非很好，由此可见，《复仇者联盟4》这部电影好评居多。



**图 2‑2-2 分别对’好评’,’一般’,’差评’绘制的词云图**

# 对发表短评的时间进行分析，探讨发表时间变化情况，分析不同城市的评分状况

**任务描述**

我们需要随着发表短评时间的变化，其数量如何变化，以及其评分的变化情况。这样我们可以很好的看出在哪些时间段用户会更活跃于《复仇者联盟4》。以及随着时间的推移，用户的评分状况会如何变化。

其次我们还要探讨用户不同城市里的评分状况，这样可以很好的观察到其城市的人文情怀，比如外国的人很可能喜欢这类型的科幻片，或者说北京人更喜欢这类型的英雄救世片等等。

**任务分析**

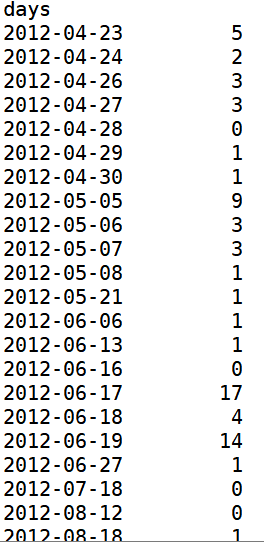
1. 分析用户发表短评数量随日期的变化情况。
2. 分析随日期变化，评分变化情况。
3. 分析用户长居城市分布情况。

## 分析用户发表短评数量随日期的变化情况

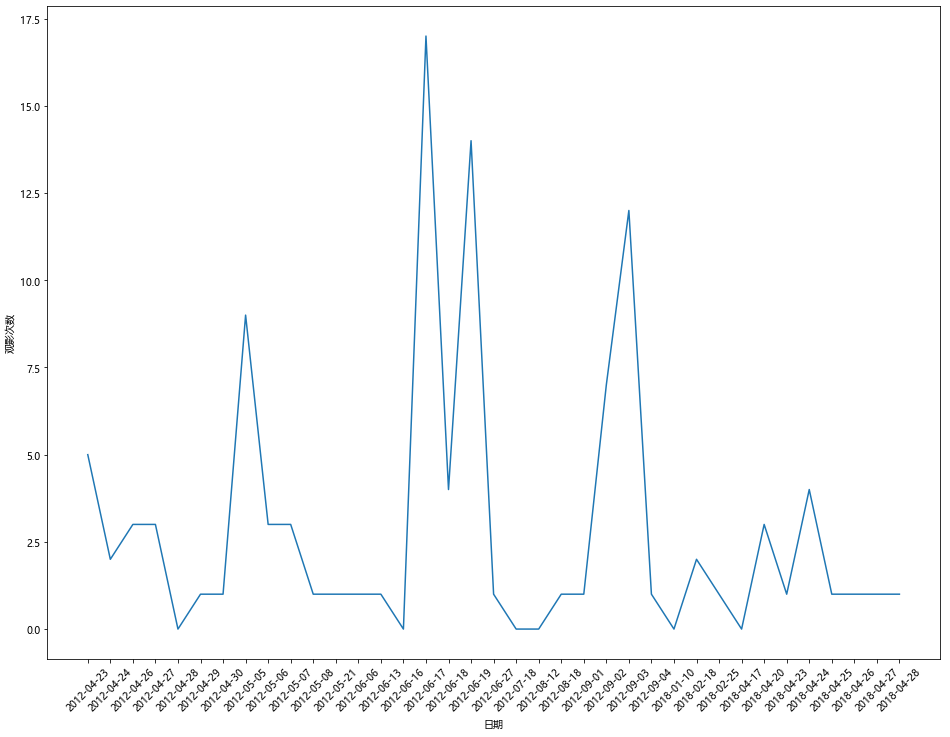
首先，我们发表短评时间列的时间是精确到秒的，并不适合我们用来统计数据，因此我们需要得到其精确到天的数据，所以我们先将发表短评时间列格式化成精确到哦天的时间作为新增一列加入数据框中，然后对新增的时间列进行组合，然后对短评数量进行统计。具体代码如下

|  |
| --- |
| # In[ TASK3 时间分析]  # 分析用户发表短评数量随日期的变化情况  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  data = pd.read\_csv(r'../01.中间数据（如有）/comments\_info1.csv',index\_col=0)  data['days'] = pd.PeriodIndex(pd.to\_datetime(data['times']),freq='D')  a = data.groupby('days').agg({'comments': 'count'})  plt.figure(figsize=(16,12))  plt.plot(range(len(a)),a)  plt.xlabel('日期')  plt.ylabel('观影次数')  plt.xticks(range(len(a)),a.index,rotation=45)  plt.show() |

由以上代码分析，得到以下数据：



直接看数据并不直观，然后我们将其折线图绘制呈现，如图3.1



**图 3-1** **短评数量随日期的变化情况图**

由图可以看出，评论数在17号和18号达到高峰，在其它日子里数量也不低。

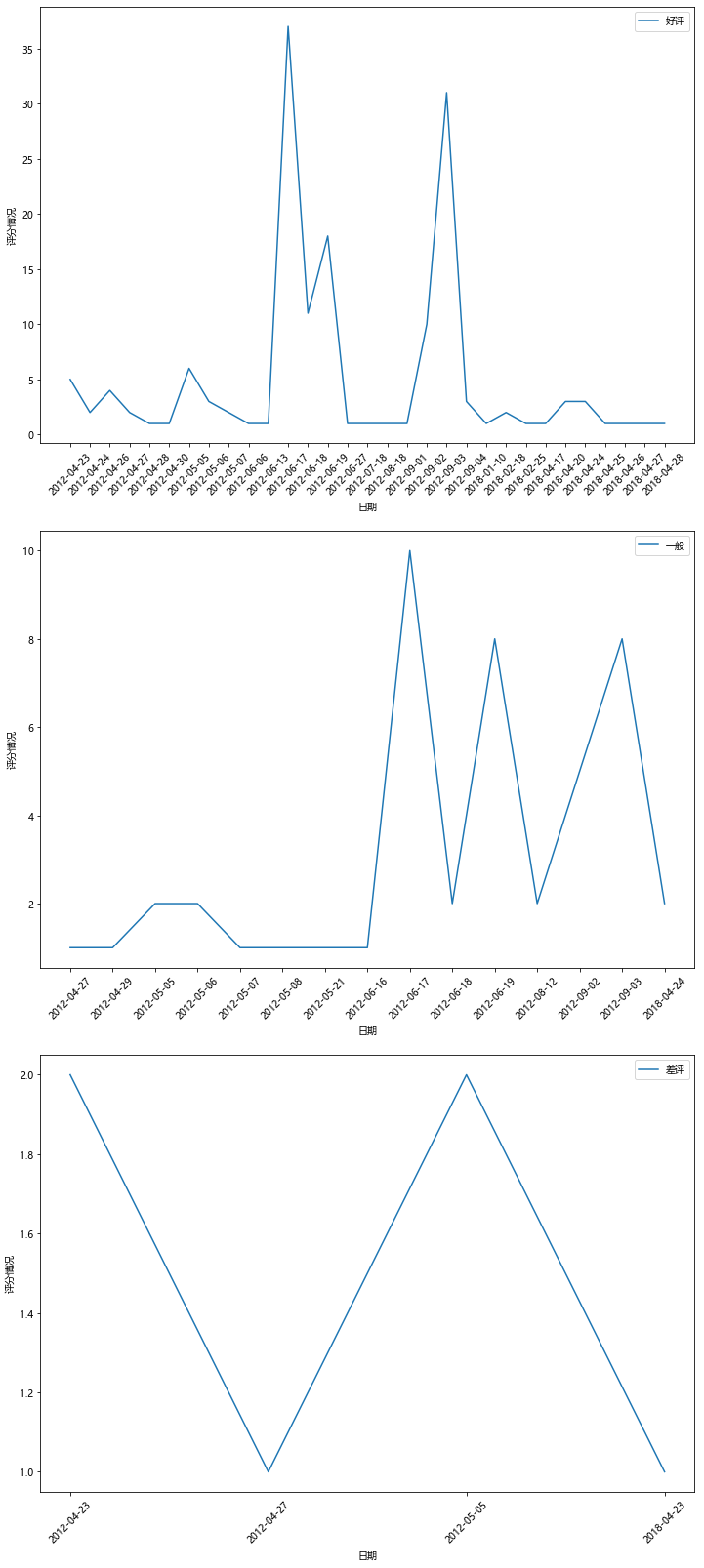
## 随日期变化，评分变化情况

除了查看短评数量的变化情况，我们还需分析其评分变化情况，这样才清楚哪些随着时间推移，大众的审美是如何变化的，或者可以看出，是否有水军，只是为了提高电影的口碑而给的好评，或是为了抹黑电影给的差评。

具体代码如下：

|  |
| --- |
| # 分析随日期变化，评分变化情况  a = data.groupby(['days','leval'],as\_index=False).agg({'username': 'count'})  fig = plt.figure(figsize=(12,28))  #好评随时间变化情况  ax1 = fig.add\_subplot(3,1,1)  ax1.plot(range(len(a.loc[a['leval']=='好评','username'])),a.loc[a['leval']=='好评','username'])  plt.xlabel('日期')  plt.ylabel('评分情况')  plt.xticks(range(len(a.loc[a['leval']=='好评','username'])),a.loc[a['leval']=='好评','days'],rotation=45)  plt.legend(['好评'])  #一般随时间变化情况  ax2 = fig.add\_subplot(3,1,2)  ax2.plot(range(len(a.loc[a['leval']=='一般','username'])),a.loc[a['leval']=='一般','username'])  plt.xlabel('日期')  plt.ylabel('评分情况')  plt.xticks(range(len(a.loc[a['leval']=='一般','username'])),a.loc[a['leval']=='一般','days'],rotation=45)  plt.legend(['一般'])  #差评随时间变化情况  ax3 = fig.add\_subplot(3,1,3)  ax3.plot(range(len(a.loc[a['leval']=='差评','username'])),a.loc[a['leval']=='差评','username'])  plt.xlabel('日期')  plt.ylabel('评分情况')  plt.xticks(range(len(a.loc[a['leval']=='差评','username'])),a.loc[a['leval']=='差评','days'],rotation=45)  plt.legend(['差评'])  plt.show() |

这里我们分别对‘好评’，‘一般’，‘差评’三个评分等级进行分析，绘图分析结果如图3.2所示。



**图 3-2 ‘好评’，‘一般’，‘差评’随日期的变化情况图**

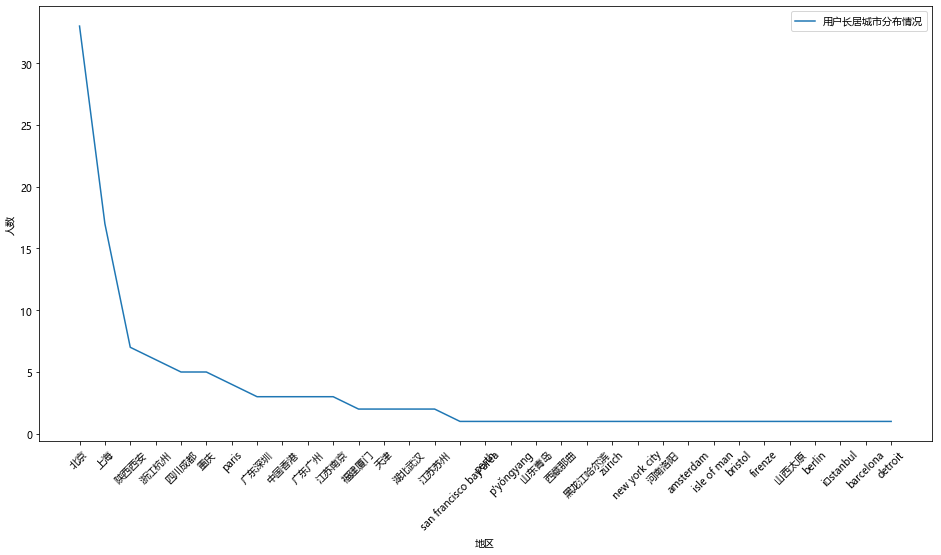
## 分析用户长居城市分布情况

评论的情况还可能受到地方人文情怀的影响，比如某个地方比较喜爱英雄主义的的片子，那么其好评数就会相对较多，因此我们还要分析发表短评的用户长居城市分布情况，从而探讨其人文情怀。

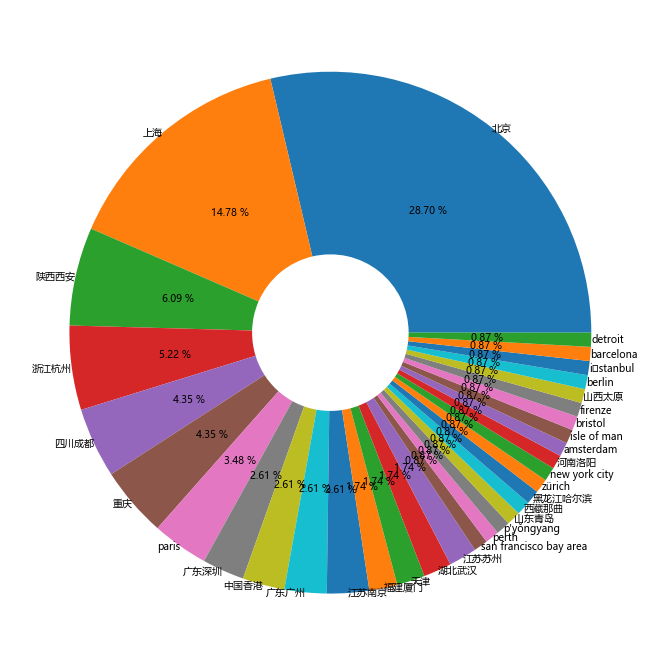
这里我们简单的对‘citys’这一列进行统计，使用pandas的value\_counts()方法即可，但由于有些用户的城市并没有给出，因此我们要排除那些没有给出城市的用户。具体代码如下。

|  |
| --- |
| import matplotlib.pyplot as plt  import pandas as pd  # 分析用户长居城市分布情况  a = data.loc[data['citys'] != '','citys'].value\_counts()  #画出折线图  plt.figure(figsize=(16,8))  plt.plot(a.index,a.values)  plt.xlabel('地区')  plt.ylabel('人数')  plt.xticks(rotation=45)  plt.legend(['用户长居城市分布情况'])  plt.show()  #画出饼图  plt.figure(figsize=(12,12))  plt.pie(a.values,autopct='%.2f %%',radius=1,labels=list(a.index),labeldistance=1,wedgeprops={  'width':0.7  })  plt.show()  # 分析不同城市的评分情况  a = data.loc[data['citys'] != '',:].groupby(['citys','leval'],as\_index=False).agg({'username': 'count'})  plt.figure(figsize=(16,8))  plt.plot(a.loc[a['leval']=='好评','citys'],a.loc[a['leval']=='好评','username'])  plt.xlabel('城市')  plt.ylabel('好评数量')  plt.xticks(rotation=45)  plt.legend(['好评数量'])  plt.show() |

我们分别对该数据画出了折线图和饼图，能够更清楚对比观测哪些地点的居民相对较多。图3.3.1为居民城市变化图，图3.3.2为占比图。



**图3-3-1 居民城市变化图**



**图3-3-2 占比图**

由图可知，显然北京和上海的用户评论更多，同时还有少部分的国外评论者。

# 用户会龄的评分影响

**任务描述**

用户会龄对评分也是具有很大的影响的，因为如果是一些很老的会员，他们给出的评价很可能就是真心的，但如果是一些刚注册不久的的人评论的话，一样很可能是水军，不太具有真实性。因此我们要对用户会龄进行绘图分析。

**任务分析**

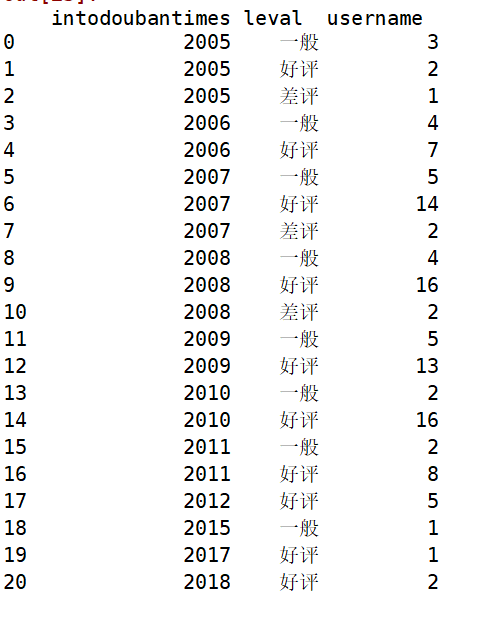
1. 分别对‘好评’，‘一般’,’差评’的评分等级统计其用户会龄。

## 分别对‘好评’，‘一般’,’差评’的评分等级统计其用户会龄

首先我们一样是要将一些没有用户会龄（没有爬取到）的用户排除掉，然后根据我们的日期获取其年份作为新增的一列加入数据框中，最后根据会员年份和和评分等级进行分组，统计用户的数量。以下是具体代码。

|  |
| --- |
| # In[TASK5 用户会龄分析]  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  #分析《复仇者联盟4》发布短评用户的会龄分布  plt.figure(figsize=(12,12))  a = pd.DatetimeIndex(pd.to\_datetime(data['intodoubantimes'].dropna())).year.value\_counts()  plt.pie(a.values,autopct='%.2f %%',radius=1,labels=list(a.index),labeldistance=1,wedgeprops={  'width':0.7  })  plt.show()    # 分析会龄对评分的影响  a = data[['intodoubantimes','leval','username']].dropna()  a['intodoubantimes'] = pd.DatetimeIndex(pd.to\_datetime(a['intodoubantimes'])).year  a = a.groupby(['intodoubantimes','leval'],as\_index=False).agg({'username': 'count'}) |

我们得到的统计结果如下图4.2.1。

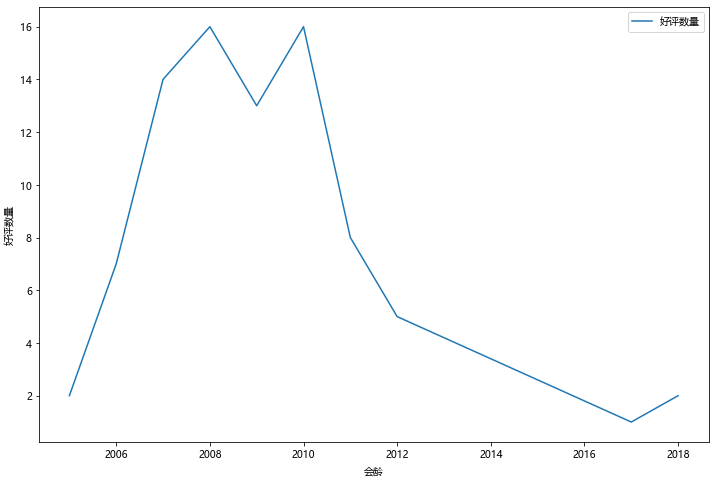


**图4-2-1评分人数随年份变化数据**

最后我们需要对好评的评分人数随时间变化的趋势绘制折线图。我们可以更直观的看出有多少老会员和多少新会员参与其中。代码如下。

|  |
| --- |
| #好评  plt.figure(figsize=(12,8))  plt.plot(a.loc[a['leval']=='好评','intodoubantimes'],a.loc[a['leval']=='好评','username'])  plt.xlabel('会龄')  plt.ylabel('好评数量')  plt.legend(['好评数量'])  plt.show() |

绘图结果如图4.2.2



**图4-2-2 评分人数随年份变化图**

显然我们可以看出会龄比较大的人数还是居多的，因此，可以知道评论没有多少水军。

# 小结

本案例通过爬取豆瓣短评的数据，并对短评数据，时间，地点等因素作分析，很好的锻炼了，我们的爬虫能力，以及我们数据分析能力。