文章编号:1007-1423(2017)35-0055-04 DOI:10.3969/j.issn.1007-1423.2017.35.011

基于 Python 爬虫的电影评论情感倾向性分析

涂小琴

(云南师范大学文理学院,昆明 650222)

摘要:

通过对豆瓣网站评分高(9.1 分)的与评分(5.2 分)两部电影进行电影评论的搜集,利用 Python 网络爬虫获取这些评论数据并清理。利用 PMI 算法,对 TF-IDF 算法进行改进,并对评论进行分类,得出 PMI 最高的 15 个分词,最后对分词进行分析统计,得出分析结果。

关键词:

Python; 爬虫; 情感分析; 影评

0 引言

随着现代生活水平的提高,更多的愿意将时间和精力投入至精神生活,越来越多的人去到电影院,作为国内电影最有影响的莫过于豆瓣(www.douban.com),豆瓣给电影爱好者提供了一个很好的分享及评论平台。每一部电影,在豆瓣上都可以看到很多的评论,可见人们对电影的喜爱。那什么样的电影是大众所喜欢的呢? 什么样的电影又是不被大众所喜爱呢,通过 Python 爬虫,我们可以获取大量的影评来进行分析,从而得出观众的喜好。

情感分析是一种常见的自然语言处理(NLP)方法的应用, NLP 情感分析中一般有两种方法,第一种是根据语义和依存关系来量化文本的情感色彩。但这种方法首先需要很完善的情感词库,另外需要很好的语言学基础,也就是说需要知道一个句子通常在什么情况为表现为 Positive 和 Negative。个人认为,我们永远无法穷尽所有的语法规则和感情词汇,这也就无形之中增加了构造分类规则的难度。第二种方法,就是基于机器学习的方法。基于机器学习,本质上就是要转化为机器学习能解决的问题。情感分析实际上就是认为是机器学习中的二分类问题。但是机器是无法理解文本的,所以我们必须能够实现让文本转化为向量,从而让机器能够理解。

通过豆瓣网站中最近热点的两部电影《摔跤吧! 爸爸》,以及电影《悟空传》的影评进行数据处理,通过 PLN 机器学习进行情感分析,分析观众情感倾向。

1 数据获取

豆瓣上有很多关于每一部电影的评论,对于一些评分比较高的电影而言,有的评论在几十万条,所以采用 Python 网络爬虫来进行处理,由于豆瓣是一个反爬虫的网站,所以在做网络爬虫时,还需要进行浏览器访问模拟,通过对网页源代码的分析,用正则表达式来获取所需数据。为了能够更好地分析观众的情感,在获取数据时,分别获取了豆瓣网站上得分高于 9.0 的两部电影,以及得分低于 4.0 的两部电影的影评数据进行分析。通过高分的两部电影与低分的两部电影进行对比分析,更能反映观众的喜好。

要获取相应的数据,不同的网站对应着不同的正则表达式,最近在豆瓣网站中得分比较高的其中一部电影为《摔跤吧,爸爸!》,评分在 9.1 分,首先通过进入豆瓣网站进入该电影影评,获取影评首页的 URL,再与影评的下一页进行 URL 比较,找到这部电影所有影评的 URL 规则,利用循环来进行读取每页影评的数据。借助库文件 Beautifulsoup、Requests 以及 Re 来进行数据的爬取,爬取的数据有:评论人、评论的时间、评论等级、评论具体内容等,因为评论量比较大,所以将获取

到的数据存放到 Excel 中。获取影评的关键代码如下：

```
get_url = url  #url 即需要获取数据的网页地址
data=requests.get(get_url, timeout =20, headers = header,cook-
ies=cookies).text
soup = BeautifulSoup(data,'lxml')
comments=soup.find_all("div", class_='comment-item')
for i in comments:
    listinfo=[]
    com=i.find('span',class_='comment-info')
    listinfo.append(com.contents[1].text.strip())  #用户名
    #rating
    rating=i.find('span',class_="rating")
    if rating!=None:
        rating=rating.get('title')
    else:
        rating='无评分'
    #print(rating)
    listinfo.append(rating)
    listinfo.append(i.find('span',class_ ="comment- time").text.
strip())  #评论时间
    #listinfo.append(i.find('span',class_ ="votes  pr5").text.strip
()) #点赞人数
    listinfo.append(i.find('p').text.strip())  # 相应的评论
    #print(i.find('p').text)
    Infolist.append(listinfo)
```

数据获取结果如图 1 所示。

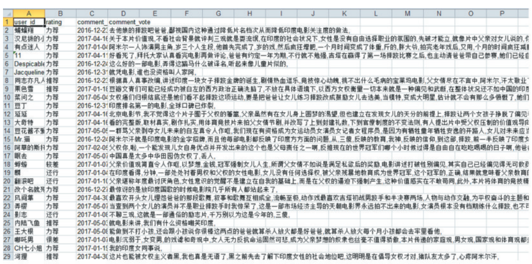


图1 获取数据结果

2 数据处理

通过图 1 可以看出，获取到的数据中有很多的语气词，如“啊”，“哦”“的”等等，还有不少的标点符号，这些都不利于我们对关键词出现次数的统计，所以需对相应的数据进行清理工作，借助“WordStop.txt”停用词来进行处理，将停用词进行对比过滤。得到清理后的数据。通过 Python 中的正则表达式模块 RE 来进行清理工作，关键代码如下：

```
pattern = re.compile(r'[\u4e00-\u9fa5]+')
filterdata = re.findall(pattern, lnfolist)
cleaned_comments = ''.join(filterdata)
```

并对数据进行停用词去除：

```
stopwords=filterdata.read_csv("stopwords.txt",index_col=False,
quoting=3,sep="\t",names=['stopword'], encoding='utf-8')
words_df=words_df[~words_df.segment.isin(stopwords.stopword)]
words_stat=words_stat.reset_index().sort_values(by=["计数"],as-
cending=False)
```

以及对词频进行统计。

3 情感分析

3.1 分类算法改进

词频(Term Frequency，TF)指的是某一个给定的词语在该文件中出现的频率。这个数字是对词数(term count)的归一化，以防止它偏向长的文件。(同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否。)对于在某一特定文件里的词语来说，它的重要性可表示为：

$$tf_{i,j}=\frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

在以上式子中，$n_{i,j}$ 是该词在文件 d_j 中的出现次数，而分母则是在文件 d_j 中所有字词的出现次数之和。

$$\mathrm{idf}_i=\log\frac{|D|}{\left|\left\{j:t_i\in d_j\right\}\right|} \tag{2}$$

其中，$|D|$:语料库中的文件总数，$\left|\left\{j:t_i\in d_j\right\}\right|$:包含词语 t_i 的文件数目(即 $n_{i,j}\neq 0$ 的文件数目)，如果该词语不在语料库中，就会导致被除数为 0，因此一般情况下使用 $1+\left|\left\{j:t_i\in d_j\right\}\right|$，然后，$tf\mathrm{idf}_{i,j}=tf_{ij}\times\mathrm{idf}_i$ 。

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的 TF-IDF。因此，TF-IDF 倾向于过滤掉常见的词语，保留重要的词语。为了能够分类出最有效的情感词，引用 PMI(Pointwise Mutual Information)，PMI 用来衡量两个事物之间的相关性(例如两个词)。所以对公式(2)进行修改。

$$PMI(word,pos)=\log\frac{P(word,pos)}{P(pos)*P(word)} \tag{3}$$

其中，pos 表示文档的情感，word 表示某一个词。

为了解决分词和词频问题，使用 Python 中的 jieba 分词库。

3.2 分词与词频的实现

在影评中的一些词，可能是常用词，所以需要把他们放入到常用词文档中，在进行分词时，就需用进行提前剔除，而有一些词语并不是普通意义上的情感词，所以需要通过公式(3)来构造这些词语的词向量。具体实现的关键代码如下：

```
def process_data(self, pos_file='./corpus/pos.txt', neg_file='./corpus/neg.txt', feature_num=30):
    stopwords=self.loan_stopwords()
    pos=self.loan_txt(pos_file)
    neg=self.loan_txt(neg_file)
    pos_seg_list=self.get_seg_list(pos, stopwords)
    neg_seg_list=self.get_seg_list(neg, stopwords)
    pos_freq_dist=self.get_freq_dist(pos_seg_list)
    neg_freq_dist=self.get_freq_dist(neg_seg_list)
    pos_words_set=self.get_words_set(pos_seg_list)
    neg_words_set=self.get_words_set(neg_seg_list)
    pos_words_num=self.compute_words_num(pos_seg_list)
    neg_words_num=self.compute_words_num(neg_seg_list)
    total_seg_list = []
    for each in pos_seg_list:
        total_seg_list.append(each)
    for each in neg_seg_list:
        total_seg_list.append(each)
    pos_words_PMI=self.PMI(pos_words_set,
        pos_seg_list, len(pos_seg_list) /
        len(total_seg_list))
    neg_words_PMI=self.PMI(neg_words_set,
        neg_seg_list, len(neg_seg_list) /
        len(total_seg_list))
    pos_words_PMI=self.sort_by_value(pos_words_PMI)
    neg_words_PMI=self.sort_by_value(neg_words_PMI)
    self.features=self.feature_list(pos_words_PMI,
        feature_num, neg_words_PMI, feature_num)
    self.save_feature_model(self.features)
    self.create_train_csv(self.features,pos_freq_dist,
        neg_freq_dist)
```

4 对比分析

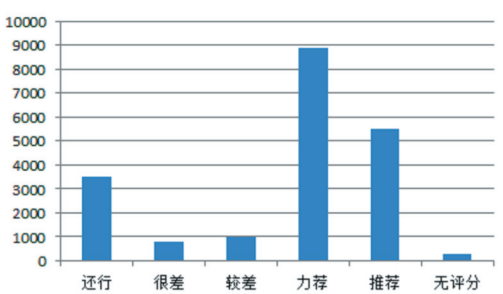
通过对两部电影影评的结果进行分析，发现，观众对两部电影的推荐度如图 2 所示。

通过对词频进行分析，得出两部电影的 PMI 最大的前 15 个词。如表 1 所示：

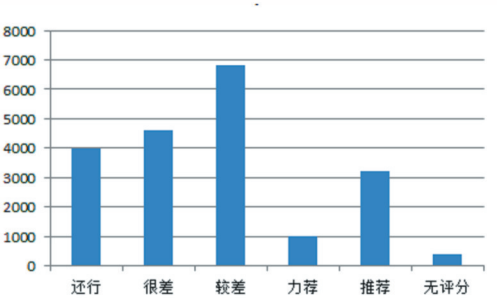
表 1

《摔跤吧！爸爸》影评结果		《悟空传》影评结果	
1	电影	1	电影
2	印度	2	特效
3	摔跤	3	孙悟空
4	父亲	4	故事
5	父权	5	没
6	热血	6	倪妮
7	励志	7	空洞
8	女权	8	垃圾
9	好看	9	彭于晏
10	成功	10	人物
11	题材	11	失望
12	爸爸	12	西游
13	爱国	13	差
14	价值观	14	猴子
15	阿米尔	15	烂片

从上表可以看出，对于《摔跤吧！爸爸》这部电影，很多评论显示的是褒义词居多，而《悟空传》评论中贬义词居多，也反映了观众的喜好心理。



《摔跤吧！爸爸》影评结果



《悟空传》影评结果

图 2

5 结语

本文通过机器学习的方法，让文本转换为向量，方便机器识别，采集的数据在 2 万条左右，数据越多，所得到的分词的权重也就越准确，文中通过 Python 语言获取大量的影评数据，将两部电影的影评进行对比分析，从 PMI 的结果可以看出来，观众更喜欢看励志，题材好的作品。

参考文献：

[1]江腾蛟，万常选，刘德喜. 基于语义分析的评价对象-情感词对抽取[J]. 计算机学报，2016.4.

[2]朱琳琳，徐建. 网络评论情感分析关键技术及应用研究[J]. 情报理论与实践，2017.1.

[3]赵刚，徐赞. 基于机器学习的商品评论情感分析模型研究[J]. 信息安全研究，2017.2.

[4]崔连超. 互联网评论文本情感分析研究[D]. 山东大学，2015.

[5]蓝天广. 电子商务产品在线评论的细粒度情感强度分析[D]. 北京邮电大学，2015.

作者简介：

涂小琴（1981-），女，讲师，研究方向为程序设计方法

收稿日期：2017-09-28　　修稿日期：2017-11-30

Sentiment Analysis of Movie Reviews Based on Python Crawler

TU Xiao-qin

(College of Arts And Sciences, Yunnan Normal University, Kunming 650222)

Abstract：

Through the collection of 9.2 Movie Reviews, and collect 5.2 minute Movie Reviews to compare, uses the Python crawler to retrieve and clean up these comments, uses the PMI algorithm to improve the TF-IDF algorithm. Classifies the reviews, gets the highest 15 participles of PMI, finally, analyzes the word segmentation, and obtains the statistically, analytical results.

Keywords：

Python; Crawler; Emotion Analysis; Film Review