后浪入海——哔哩哔哩评论数据分析

余柏辰

本项目在我的个人GitHub上已开源发布，完整的代码和分支详见这里。

摘要：在互联网上，我们常常关注的一个问题即我们的用户或受众是谁？他们对接收到的讯息有怎样的态度？本项目实现了一个对哔哩哔哩视频网（以下简称b站）评论区的数据分析小程序。通过对用户输入的av号或BV号（或给定json格式的数据文件），爬取对应视频的评论区有效数据，对其进行数据清洗后进行初步的描述性统计分析。然后通过调用第三方库，实现如词云、情感分析等功能。均通过直观的图表格式给出分析结论，供相关人士参考。

问题背景

该项目的问题意识与灵感来自互联网上对哔哩哔哩官方视频《后浪》的广泛争论，我想要对b站评论区的反应进行数据分析。刚好这时b站推出了第二弹视频《入海》，于是可以进行对比分析。然而听说还可能有第三弹，于是决定将其做成一个对b站任意视频评论区进行数据分析的小工具，然后用它进行分析研究。

事实上，由于互联网或是新媒体的高度圈层化，这样的分析有助于我们认清互联网舆论的现状及形成过程。

模型建立

将爬取来的数据简单整理后保存至json文件，便于重复使用。在分析时导入DataFrame中进行操作，每一行即为单个评论的全部信息。

问题求解

注：我在编程初期采用的是多文件的脚本编码方式，这有助于模块化地发现错误和更新功能；后期由于要制作GUI接口，就直接将所有的函数放到一个文件来添加GUI功能。因此，下面的代码皆为主干代码，仅用于展示算法和编码思路，详见GitHub上分支；完整代码在这里。

数据获取

由于b站在robots.txt中明确很多不允许爬取的内容且进行了反爬处理，我们并不直接从b站视频地址上爬取数据，而是找到其API站点进行爬取。这样既便捷，数据格式也相对整齐。因此，这样的爬取只需要调取requests库的get方法即可完成，代码如下。

```
while True:
    # noinspection PyBroadException
    try:
        url = "https://api.bilibili.com/x/reply?type=1&oid={0}&nohot=1&sort=0&pn=
{1}".format(aid, j)
        html = get(url)
        data = loads(html.text)
        for i in range(0, 20):  # 每页20条评论
            timeArray = localtime(data['data']['replies'][i]['ctime'])  # 时间戳
            date.append(strftime("%Y-%m-%d %H:%M:%S", timeArray))
            level.append(str(data['data']['replies'][i]['member']['level_info']
```

```
['current_level']))  # 用户等级
            txt = data['data']['replies'][i]['content']['message']  # 评论内容
            txt = sub('\n', '', txt)
            txt = sub('\r', '', txt)
            text.append(txt)
            like.append(str(data['data']['replies'][i]['like']))  # 赞同数
        j += 1
    except Exception:
        break
```

此外，由于很多数据数据量过大（如我们提到的《后浪》《入海》，均有数万条评论），每次新爬取都要耗时数分钟，于是我们制作了已有数据导入的接口，通过设置flag标记变量以决定是否要去爬取新数据。

事实上，如果采用多线程爬虫是解决问题的一种有效路径，可以显著提高爬取速度，将在以后逐步完善。

数据处理与数据清洗

从爬取好或加载的json文件中读取数据至DataFrame结构的df数据表，在预处理时实现jieba分词并添加至DataFrame中，这样做的好处是分担后面的运行时间，提高数据分析的操作体验。

此外，由于爬取时不时会出现连续两行数据相同的情况，我们调用pandas库的数据清理方法进行简单的清洗。诚然，这样粗暴的清洗会导致如“来啦！”之类的评论减少，但这并不影响我们的分析————反而有一定好处，这意味着我们在后面制作词云时不用考虑大量在互联网上被使用却无意义的语汇。

```
df = pd.read_json(jsonfile)
df['cut'] = pd.Series(dtype=np.float64)
for i in range(0, df.shape[0]):
    text = df['text'].iloc[i]
    txt_cut = lcut(text, cut_all=False)
    txt_cut = ' '.join(txt_cut)
    df['cut'].iloc[i] = txt_cut
# data cleaning
df.drop_duplicates(inplace=True)
df.dropna(inplace=True)
```

算法和具体功能的实现

BV号与av号互转

由于b站的迅速扩张或是对Youtube的单纯模仿，他们认为原有的顺序排列的av号不能满足需求，于是将视频的默认序号改为BV号。然而，其转换算法已经被破解,即base58的算法。

我们要做到对于用户需要分析的视频，输入任意一种视频代号即可开始工作。考虑到至少目前av号依旧可以使用，API也是提供的av号版本。我们首先识别视频号的类型，若为BV号则通过下面的脚本转换为av号，便于后续处理。

```
# 作者：mcfx
def bv2av(bv):
```

```
    table = 'fZodR9XQDSUm21yCkr6zBqiveYah8bt4xsWpHnJE7jL5VG3guMTKNPAwcF'
    tr = {}
    for i in range(58):
        tr[table[i]] = i
    s = [11, 10, 3, 8, 4, 6]
    xor = 177451812
    add = 8728348608
    r = 0
    for i in range(6):
        r += tr[bv[s[i]]] * 58 ** i
    return (r - add) ^ xor
```

描述性统计

这个部分我们通过matplotlib的pyplot工具绘制一些图表，以助于我们对该视频评论区有初步认识。

第一个图是用户等级饼状图，用以分析视频的受众，尤其是那些被其吸引或争论而评论的观众。

这里要普及一下b站的等级制。b站有0-6级共七个等级，与用户权限直接关联。而不同于QQ等软件对等级的无限制，b站保持了较严格的水准。一般情况下，6级号或者是入站时间长、了解很多相关知识的老用户，要么是某一领域的优秀up主，或者是氪金用户。而0级号则是注册而未通过答题的（近年来题目已经日益简单）。因此，对b站不同等级用户的观察就显得很有意义。如生活区的讨论质量就普遍低于动漫区（与讨论门槛有关）。

```
def pie(data, ax):
    # 用户等级饼状图—视频受众分析
    x = data['level'].value_counts().sort_index(ascending=True)
    ax.pie(x, labels=list(x.index), startangle=180, shadow=True,
autopct='%1.2f%%')
    ax.set(title="评论等级分布")
    ax.grid()
```

后两个图分别为等级——点赞散点图和等级时序图，主要关注谁在主导舆论。我们常说互联网拉近了人与人的距离或使人更加平等，是这样吗？事实上，互联网往往复制甚至放大了现实社会中的关系。先来的人，有时并不只是因为碰巧刷到而获得较多点赞；而等级低的评论也很难浮在上面。我们会在后面的数据分析章节中给出实例。

```
def scatter(data, ax):
    x = data['level'].sort_index(ascending=True)
    y = data['like'].sort_index(ascending=True)
    ax.scatter(x, y, marker='.')
    ax.set_title("等级—点赞散点图")
    ax.grid()
```

```
def timeseries(data, ax):
    x = data['date']
```

```
    y = data['level']
    ax.plot_date(x, y, )
    ax.xaxis.set_major_formatter(DateFormatter('%Y-%m-%d'))
    ax.xaxis.set_major_locator(AutoDateLocator(maxticks=8))
    ax.legend(['评论'], loc='upper right')
    ax.set_title("用户等级时序分析")
    ax.set_ylabel("等级")
    ax.grid()
```

制作词云

基于前面通过jieba处理好的数据，我们通过第三方库wordcloud生成词云。其中用到了Pillow库以创建背景图（轮廓外需为全白），然后通过我们熟悉的matplotlib库中的pyplot工具弹窗生成词云图，这在我们的GUI编程中显得恰如其分。

```
def wordcloud(series):
    text_cut = ""
    for i in series:
        text_cut += i
    stop_words = open(r"data/stopwords.txt", encoding='utf-8').read().split('\n')
    background = Image.open(r"data/iconimage.png")
    graph = np.array(background)
    word_cloud = WordCloud(font_path="C:/Windows/Fonts/simfang.ttf",
background_color="white", mask=graph,
                           stopwords=stop_words)
    word_cloud.generate(text_cut)
    plt.subplots(figsize=(12, 8))
    plt.imshow(word_cloud)
    plt.axis("off")
    plt.savefig("clouds.png",dpi=300)
    plt.show()
```

情感分析

接下来进入了自然语言处理（NLP）的领域。这里先介绍一下情感分析的步骤。由于中文的特殊性，我们首先要进行分词，然后划分数据集，将文本变为电脑可以识别的词向量，然后喂入分类器即可。

我们这个版本采用的是第三方库snownlp，然而相比于我们前面用到的jieba，它的分词性能较差。由于其采用的是Naive Bayes分类器，限于能力我也很难在这个版本中给出更好的优化（如果只是单纯地用jieba分词替代效果还不如已经造好轮子的snownlp），因此这也是这个版本的一大遗憾。

```
def feeling():
    global df
    df['sentiments'] = pd.Series(dtype=np.float64)
    df['result'] = pd.Series(dtype=str)
    for i in range(0, df.shape[0]):
        text = SnowNLP(df['text'].iloc[i])
        df['sentiments'].iloc[i] = text.sentiments
```

```
        if text.sentiments > 0.7:
            df['result'].iloc[i] = "积极"
        else:
            df['result'].iloc[i] = "消极"
```

上面展示的代码仅为sentiments列的构造过程，我们还使用matplotlib进行了可视化展示，包括饼状图和时序图。饼状图可以帮我们清晰地看到两方人数比，且后续可以通过Echarts加入分等级下钻饼状图的功能，以观察不同人群用户的态度；而时序图则可以清晰地表明何种舆论在何时"入场"，在后面的实例分析中会举例说明。

但我们也意识到了深度学习的方法，可以通过搭建RNNs下的LSTM模型大幅度地改善效果。但这也存在弊病，即能否找到可以适应所有类型评论的数据集——目前至少比不过snownlp。因此，若想实现情感分析的有效改进并非易事，希望在后续版本可以做到。

GUI接口设计

作为一个脚本函数拼凑而成的小工具，实无必要大动干戈请来PyQt这样的大家伙，简单的tkinter已经足以。当然，为了美观（与新系统接轨），我们也调入了tkinter库中的ttk，使之更现代化。

 GUI界面图

在具体的GUI设计中，我们将界面分为上下两个Frame，frm_top包括一个文本框、用以接收用户输入视频号的单行输入框以及确认按钮。frm_bottom则包括四个按钮，功能分别为导入数据、生成词云、用户分析、情感分析。其余三个按钮实现的功能前面已经介绍过，这里不再赘述。而其中"导入数据"按钮通过filedialog.askopenfilename方法获取用户想要导入的数据并传至前面的datapre函数中，获得对应的DataFrame。

```
def dataFind():
    root = Tk()
    root.withdraw()
    filepath = filedialog.askopenfilename()
    global flag
    flag = False
    datapre(filepath)
    flag = True
```

此外，程序还设计了菜单栏，包括获取目录功能、程序功能介绍及帮助和原创声明等。

小结

本章节主要介绍了我编写这个项目的主要方法和功能，其中也有一些缺憾，或者说展望吧。正如我在项目摘要中说的那样，这只是个开始，希望以后可以将这个项目填充的更加丰富。

1. 挖掘更多的描述性统计图表
2. 大规模的API爬取耗时过长，可以通过多线程爬虫高速爬取
3. 自己构建更准确的分类器，以实现更好的情感分析效果
4. 可以对比不同平台（如网易云音乐）评论区的热词频率，进而分析不同软件受众的特点

数据分析实例