

微博热搜分析



目录

1 实验引言	2
1.1 研究背景	2
2 问题描述	2
2.1 研究题目	2
2.2 研究难点	2
3 实验方法	3
4 实验过程	3
4.1 数据获取	3
4.2 数据预处理	6
4.2.1 剔除广告	6
4.2.2 部分处理时剔除每天最后五十条	7
4.3 数据可视化	8
4.3.1 词云 wordcloud	8
4.3.2 每日关键词 top20	10
4.3.3 动态柱状图	13
4.3.4 情感分析	14
4.3.5 多项式拟合	17
5 实验评价	18
6 后续工作	19
7 结论	19

1 实验引言

1.1 研究背景

2022 年 12 月中旬，各类新闻层出不穷，本身就处于信息传播迅速的时代，恰逢这届不断爆冷的世界杯进入决赛阶段，阿根廷万众瞩目，又赶上国内开放的浪潮，消息更是眼花缭乱。于是突发奇想，想把信息整合可视化处理一下，以此更好地探究、观察并记录时下热点。毕竟信息爆炸的时代，消息太多了，就以微博为例，一天能有三四百条不一样的热搜，很多上榜时间短，尚未来得及看到就掉了或者被压了，所以想借此机会，记录没有记忆的互联网，并对每天的新闻有一个大致的把握。

新浪微博目前是国内第一的社交媒体，热搜榜每分钟实时更新，而微博提出的“最新鲜、最热门、最有料”也使其成为用户心中毫无疑问的 top 热搜榜。因此，该项目选择了新浪微博这一平台作为数据来源，专门对其热搜榜上的数据进行爬取与分析。

2 问题描述

2.1 研究题目

总的来说，该项目首先是爬取微博热搜榜数据，再着重对微博热搜进行可视化处理，做成柱状图、动态柱状图和折线图等图表，还有词云，并做相关情感分析，来看网上舆论的情感倾向，同时结合词云做一个对比。最后结合相关信息做一个热搜走势的拟合，来探究走势并预测其他未知热搜的潜力。

2.2 研究难点

- ① 数据获取：自学爬虫。开始有点艰难，完全看不懂 html，后面逐渐上手，还是实践出真知，且手机模式加密少，更容易。
- ② 可视化：词云制作里有很多停用词，还有一些词语划分要手动调整，动态柱状图的布局也需调整。

- ③ 拟合：一开始想最小二乘来拟合，最后试了很多，还是用了 numpy 库里的 polyfit 来处理，处理得比较粗糙，重点放在了可视化展示。

3 实验方法

- ① 在 Windows 环境下，利用 vscode (python 3.10.2) 和 jupyter notebook (python 3.9.12) 来挖掘和可视化数据，用 Excel 来存储 csv 数据。数据挖掘时，用任务计划程序来定时重复执行爬虫代码，获取原始数据。
- ② 调用 pyecharts, plotly 和 matplotlib 等库来进行数据可视化，jieba 库来分词处理文本，snownlp 进行情感分析，wordcloud 制作词云。

4 实验过程

4.1 数据获取

首先导入需要用到的库，

```
1. import requests # 向页面发送请求
2. import csv      # 写入 csv 文件
3. from bs4 import BeautifulSoup as BS # 解析页面
4. import time     # 记录爬虫的时间
```

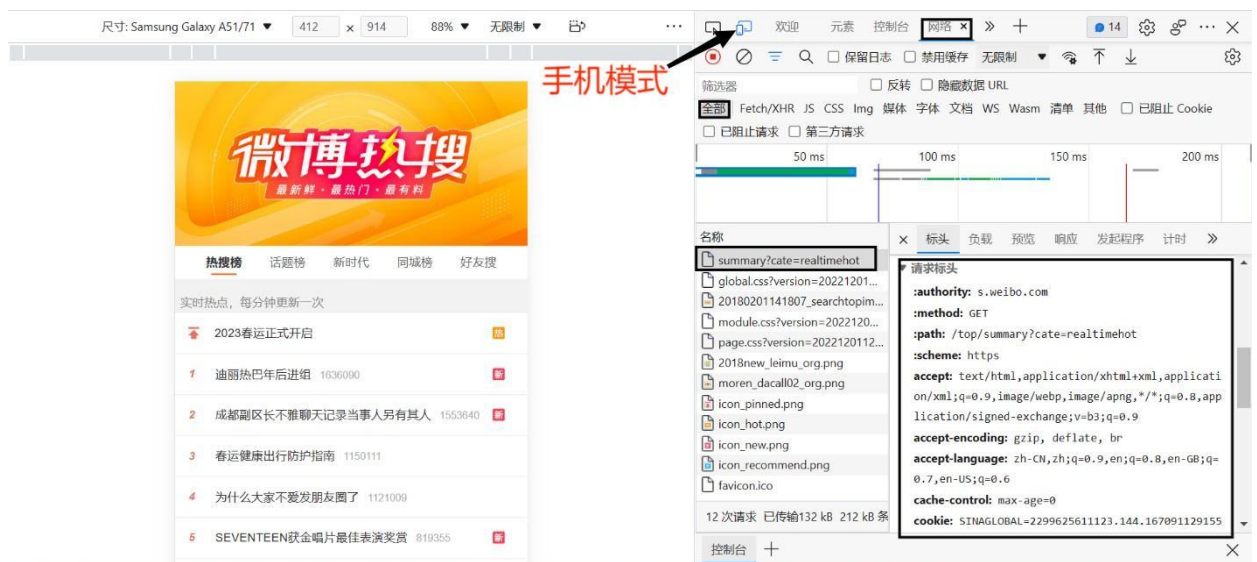
再定义一个爬取的目标地址，

```
1. url = 'https://s.weibo.com/top/summary?cate=realtimehot'
```

然后定义一个请求头，

```
headers = {
    'User-Agent': 'Mozilla/5.0 (Linux; Android 8.0.0; SM-G955U Build/R16NW) AppleWebKit/537.36 (KHTML, like Ge
    'Host': 's.weibo.com',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,applicati
    'Accept-Language': 'zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6',
    'Accept-Encoding': 'gzip, deflate, br',
    'Cookie': 'SINAGLOBAL=2299625611123.144.1670911291558; SCF=AvG3iCW-ED4m2pzB66A7ImFVKQWSnoFyWTyyTpIV5-myioE
```

这些参数在开发者模式里可以找到，这里是用手机模式爬取的，减少加密困难。



接下来向页面发送请求并解析返回的页面。

1. `response = requests.get(url = url, headers = headers)`
2. `# print(response.status_code)` 如果返回 **200** 则正常，**404** 是未找到
3. `response.encoding = "utf_8_sig"`
4. `soup = BS(response.text, "html.parser")`

根据页面分析，每条热搜都放在了标签为 `section class = "list"` 的数据里，里面每条热搜，又是 `li` 下的一个 `a` 标签。



于是根据这个思路，再去除置顶，处理标签为空的情况。将标题、排名、热度、标签、时间组成一个小列表 list_t，放入大列表 List 中，便于写入 csv。代码如下：

```
1. div_content = soup.find('section', {'class': 'list'})
2. t = 1 # 去掉置顶
3. list_t = []
4. List = []
5. for item in div_content.find_all("li"):
6.     if(t == 1):
7.         t = 0
8.         continue
9.     rank = item.find("strong").string # 排名
10.    subject = item.span.contents[0] # 标题
11.    hotness = item.find("em").string # 热度
12.    if(item.find("i")):
13.        sign = item.find("i").get("class")[1] # 标签
14.    else:
15.        sign = 'None'
16.    current_time = time.strftime("%Y-%m-%d %H:%M")
17.    list_t = [subject, rank, hotness, sign, current_time]
18.    List.append(list_t)
```

最后将 List 写入 csv 文件：

```
1. def store(List):
2.     with open('D:/code/12_23.csv', 'a', encoding = "utf_8_sig", newline = "") as file:
3.         writer = csv.writer(file)
4.         for item in List:
5.             writer.writerow(item)
6.         file.close()
```

华西医院23岁医学生抢救无效去世				
	A	B	C	D
1	华西医院23岁医学生抢救无效去世	1	2921285	icon_new
2	法国2比0摩洛哥	2	1175558	None
3	儿童感染奥密克戎应注意这几点	3	933085	None
4	OPPO Find N2 系列发布			icon_recommend
5	杭州核酸单管每人16元	4	824289	None
6	阳了真的会浑身疼吗	5	799083	icon_new
7	同事阳了自己不去上班算旷工吗	6	748230	None
8	我的守护星			icon_recommend
9	姆巴佩进球比赛法国从未输球	7	697509	None

由于需要长时间反复爬取，无法人工完成，于是设定任务计划程序来完成：

每天早上 8 点到晚上 10 点，每隔 10 分钟执行一次程序。

常规		操作	条件	设置	历史记录(已禁用)
触发器	详细信息				状态
每日	在每天的 8:00 - 触发后，在 14:00:00 期间每隔 10 分钟 重复一...				已启用

常规	触发器		条件	设置	历史记录(已禁用)
创建任务时，必须指定任务启动时发生的操作。若要更改这些操作，使用“属性”命令打开任务属性页					
操作		详细信息			
启动程序		D:\code\INTRODUCTION.py\spider.py			

每天的热搜都保存为一个 csv 文件，该项目爬取了一个礼拜 7 天的热搜，分别保存，共 7 个 csv 文件。每次爬取包括广告和目标的五十条热搜信息，一天执行 85 次程序，预处理后共 4250 条一天。

4.2 数据预处理

4.2.1 剔除广告

413	华西医院23岁医学生抢救无效去世	1	4380431	icon_boil	2022/12/15 9:20
414	女子坐地铁不戴口罩称有病毒的人才戴	2	1947666	icon_new	2022/12/15 9:20
415	儿童感染奥密戎应注意这几点	3	1643571	icon_boil	2022/12/15 9:20
416	苏醒实现电视自由			icon_recommend	2022/12/15 9:20
417	阳了真的会浑身疼吗	4	1369589	None	2022/12/15 9:20
418	至今还未感染的你是怎么做到的	5	1221506	icon_new	2022/12/15 9:20
419	法国2比0摩洛哥	6	976020	None	2022/12/15 9:20
420	宝贝成长守护计划			None	2022/12/15 9:20
421	杨迪晒阳敌表情包	7	865106	icon_new	2022/12/15 9:20
422	中国医药与辉瑞公司签订协议	8	788825	None	2022/12/15 9:20
423	少爷和我	9	674556	None	2022/12/15 9:20
424	张杰也逃不过家长群接龙	10	588791	icon_hot	2022/12/15 9:20

微博热搜榜自带广告推荐，有的标签是 icon_recommend，有的无标签，但都不存在热度且不参与排名，所以直接剔除有空值的条目。以 12 月 15 号的数据为例：


```

7 data = pd.read_csv('days/12_15.csv', header = None, encoding = 'utf-8-sig')
8 print("处理前: %d条" % data.shape[0])
9 data = data.dropna()
10 print("剔除热度为0后: %d条" % data.shape[0])

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

PS D:\code> python -u "d:\code\INTRODUCTION.py\nlp_day.py"
处理前: 4386条
剔除热度为0后: 4250条
PS D:\code>

```

4.2.2 部分处理时剔除每天最后五十条

在制作每小时的情感分析图时，每小时截取的是 0 分到 50 分 6 个时间点，而数据集是 8 点整到 22 点整，所以处理时选择删去最晚的 22 点整，最后合并为 1 个 csv，来制作一天 14 个小时的七天情感分析图。

此处要先剔除广告后，重新编号，才能将最后五十条热搜去除，也可以根据时间删除：

```

1. def clean(f_1):
2.     f_1 = f_1.dropna()
3.     f_1 = f_1.reset_index()
4.     f_1.drop(f_1.tail(50).index, inplace = True)
5.     return f_1

```

处理前 7 个单独的 csv，共 30555 条，处理完后共有 29400 条：

```

12 f_1 = pd.read_csv('days/12_15.csv', header = None, encoding = 'utf-8-sig')
13 f_2 = pd.read_csv('days/12_16.csv', header = None, encoding = 'utf-8-sig')
14 f_3 = pd.read_csv('days/12_17.csv', header = None, encoding = 'utf-8-sig')
15 f_4 = pd.read_csv('days/12_18.csv', header = None, encoding = 'utf-8-sig')
16 f_5 = pd.read_csv('days/12_19.csv', header = None, encoding = 'utf-8-sig')
17 f_6 = pd.read_csv('days/12_20.csv', header = None, encoding = 'utf-8-sig')
18 f_7 = pd.read_csv('days/12_21.csv', header = None, encoding = 'utf-8-sig')
19
20 l = [f_1, f_2, f_3, f_4, f_5, f_6, f_7]
21 data = pd.concat(l)
22 print("处理前: %d条" % data.shape[0])
23
24 f_1 = clean(f_1)
25 f_2 = clean(f_2)
26 f_3 = clean(f_3)
27 f_4 = clean(f_4)
28 f_5 = clean(f_5)
29 f_6 = clean(f_6)
30 f_7 = clean(f_7)
31 l = [f_1, f_2, f_3, f_4, f_5, f_6, f_7]
32 data = pd.concat(l)
33 print("处理后: %d条" % data.shape[0])

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```

> python -u "d:\code\INTRODUCTION.py\nlp_hour.py"
处理前: 30555条
处理后: 29400条
PS D:\code>

```


4.3 数据可视化

4.3.1 词云 wordcloud

这一部分主要是 jieba 库分词和 wordcloud 库的词云制作。

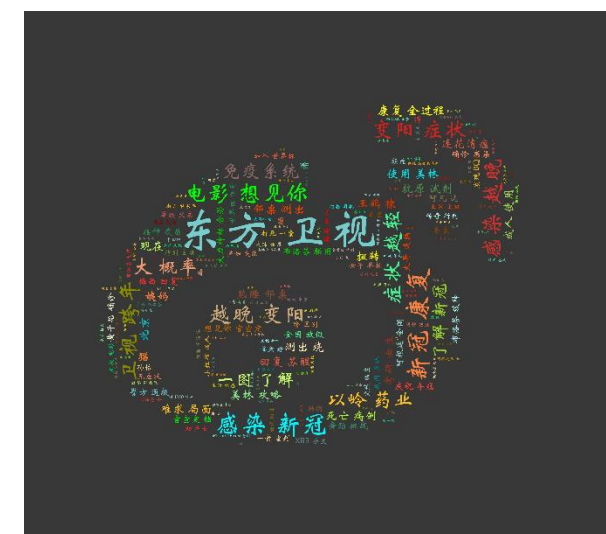
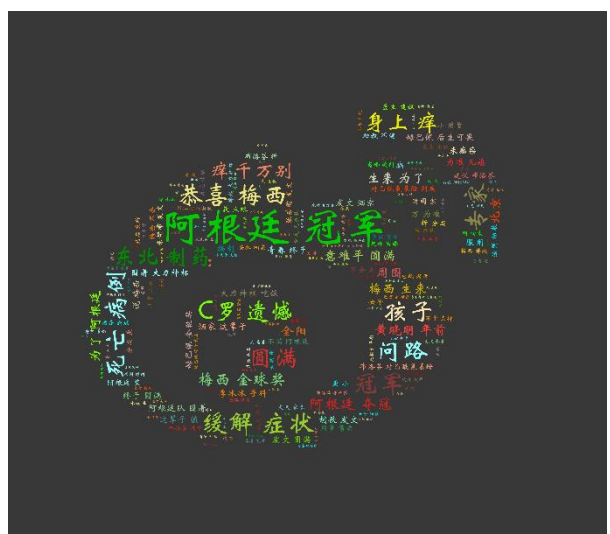
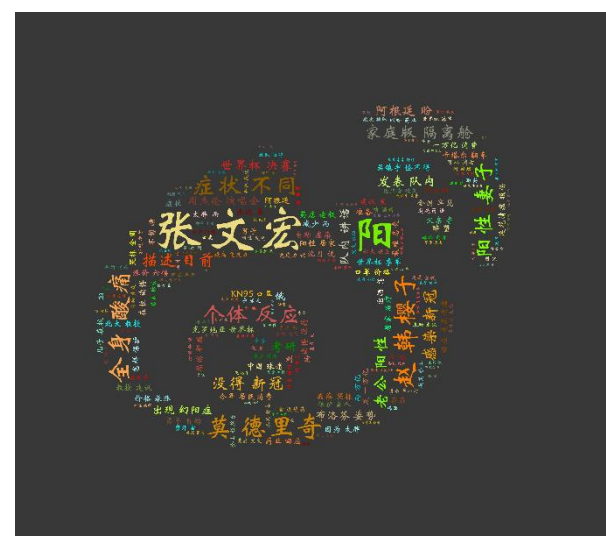
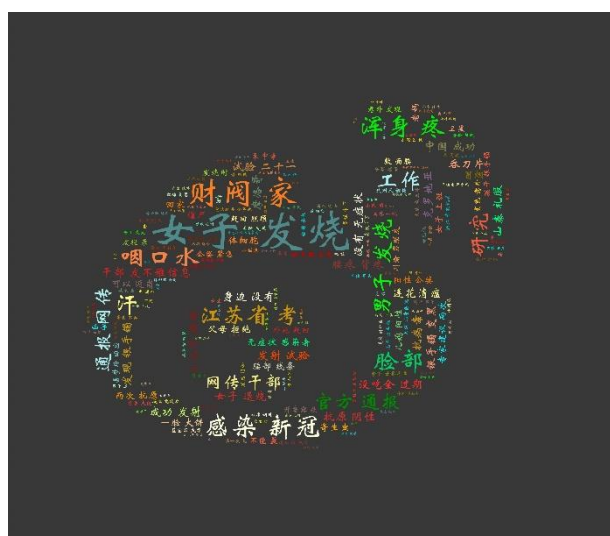
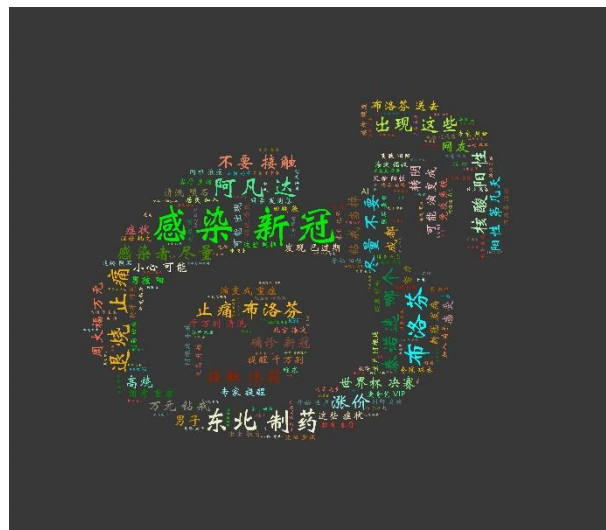
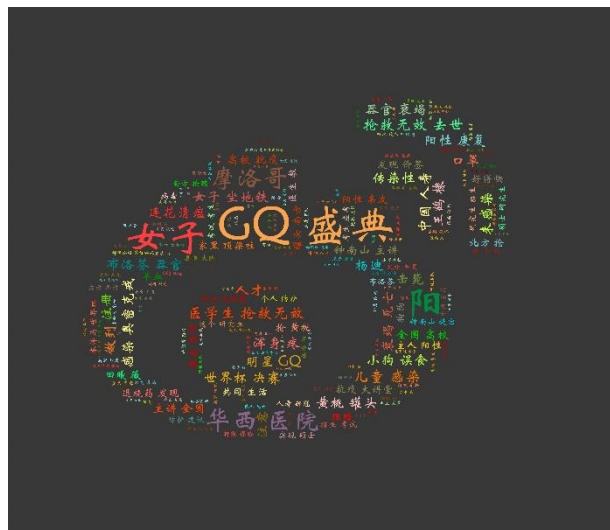
部分词语需要人工手动调整：

```
1. jieba.suggest_freq('奥密克戎', True)
2. jieba.suggest_freq('连花清瘟', True)
3. jieba.suggest_freq('吞刀片', True)
4. jieba.suggest_freq('咽口水', True)
5. jieba.suggest_freq('C 罗', True)
6. jieba.suggest_freq('发不雅信息', True)
7. jieba.suggest_freq('川渝', True)
8. jieba.suggest_freq('想见你', True)
9. jieba.suggest_freq('黄子韬', True)
10. jieba.suggest_freq('新冠', True)
11. jieba.suggest_freq('撸下口罩', True)
12. jieba.suggest_freq('姆巴佩', True)
13. jieba.suggest_freq('涮肉', True)
14. jieba.suggest_freq(('瘟', '能'), True)
15. jieba.suggest_freq(('瘟', '一次'), True)
16. jieba.suggest_freq(('期', '杀'), True)
```

也有很多停用词，这里数据量还行，所以没有查停用词表，手动输的，最后效果也不错：

```
stopwords = {'岁','的','是','吗','和','了','被','你','这','能','至今','颗',
             '怎么','对','已','称有','vs','成','都','有','多','还','一起',
             '不戴','我','比','贴','层涨','元','摔','条','上','千元','还有',
             '卖','到','应','后','戴','买','吃','天','与','却','真的','又',
             '不','将','宣布','当定','一盒','会','粒','买块','患','嫌','要',
             '点','让','一板','只','第','再','种','什么','一支','不了','干',
             '天内','知道','不到','个','为何','才','遇人','或超','薇给','超',
             '给','病','因','万买','冠','半','一次','它在','想','称','穿','疯',
             '度','就','在','从','摊','它','差点','囤','号','叫','全是','带到',
             '为什么','像','是因为','有人','开','时','月','轮','斤','小时','造',
             '毛','么','她','前','年','不是','瘟','称此','做好','哪','我们',
             '例','花','把','期间','当回事','如何','增','他们','带来','人','期',
             '杀','帮','引','或','一','名','秒','可','写','也','没','几乎',
             '地','变','忍住'})
```

制作了 7 天的词云图：





12 - 21

这些词云都用了微博标志大眼仔的蒙版，更好看一点。出现频率越高的词语在词云中显示的字号越大，此外还需加载中文本地的字体，这里用了黑体，所欲整个 word_cloud 定义是：

```
41 word_cloud = WordCloud(  
42     scale = 8,  
43     font_path = 'C:/Windows/Fonts/STXINWEI.TTF', # 显示中文需要加载本地中文字体  
44     background_color = '#383838',  
45     colormap = colormap,  
46     prefer_horizontal = 0.8, # 水平显示的文字相对于竖直显示文字的比例  
47     mask = icon, # 添加蒙版  
48     relative_scaling = 0.3, # 设置字体大小与词频的关联程度为0.3  
49     max_font_size = 80, # 设置显示的最大字体  
50     stopwords = {'岁', '的', '是', '吗', '和', '了', '被', '你', '这', '能', '至今', '颗',  
51                 '怎么', '对', '已', '称有', 'vs', '成', '都', '有', '多', '还', '一起',  
52                 '不戴', '我', '比', '贴', '层涨', '元', '摔', '条', '上', '千元', '还有',  
53                 '卖', '到', '应', '后', '戴', '买', '吃', '天', '与', '却', '真的', '又',  
54                 '不', '将', '宣布', '当定', '一盒', '会', '粒', '买块', '患', '嫌', '要',  
55                 '点', '让', '一板', '只', '第', '再', '种', '什么', '一支', '不了', '干',  
56                 '天内', '知道', '不到', '个', '为何', '才', '遇人', '或超', '薇给', '超',  
57                 '给', '病', '因', '万买', '冠', '半', '一次', '它在', '想', '称', '穿', '疯',  
58                 '度', '就', '在', '从', '摊', '它', '差点', '园', '号', '叫', '全是', '带到',  
59                 '为什么', '像', '是因为', '有人', '开', '时', '月', '轮', '斤', '小时', '造',  
60                 '毛', '么', '她', '前', '年', '不是', '瘟', '称此', '做好', '哪', '我们',  
61                 '例', '花', '把', '期间', '当回事', '如何', '增', '他们', '带来', '人', '期',  
62                 '杀', '帮', '引', '或', '一', '名', '秒', '可', '写', '也', '没', '几乎',  
63                 '地', '变', '忍住', '取决于'})  
64 word_cloud.generate(words)
```

最后保存为.jpg 文件：

```
1. word_cloud.to_file('wc1.jpg')
```

后续会在情感分析部分一起分析 16 号和 19 号的词云。

4.3.2 每日关键词 top20

词云更适合展示，有时可能有些眼花缭乱，于是又做了每日关键词 top20，

更加简洁。前面一小部分处理与词云相同，从词义分割开始不一样，top20 用了字典来做词频统计并排序：

```
1. words = jieba.lcut(str)
2. word_dict = {}
3. for word in words:
4.     if len(word) == 1: # 如果关键字字数为1，不统计；否则加1
5.         continue
6.     else:
7.         word_dict[word] = word_dict.get(word, 0) + 1
8. word_zip = zip(word_dict.values(), word_dict.keys())
9. word_sort = list(sorted(word_zip, reverse = True))# 对元组里面的元素按照 value 从大到小进行排序
```

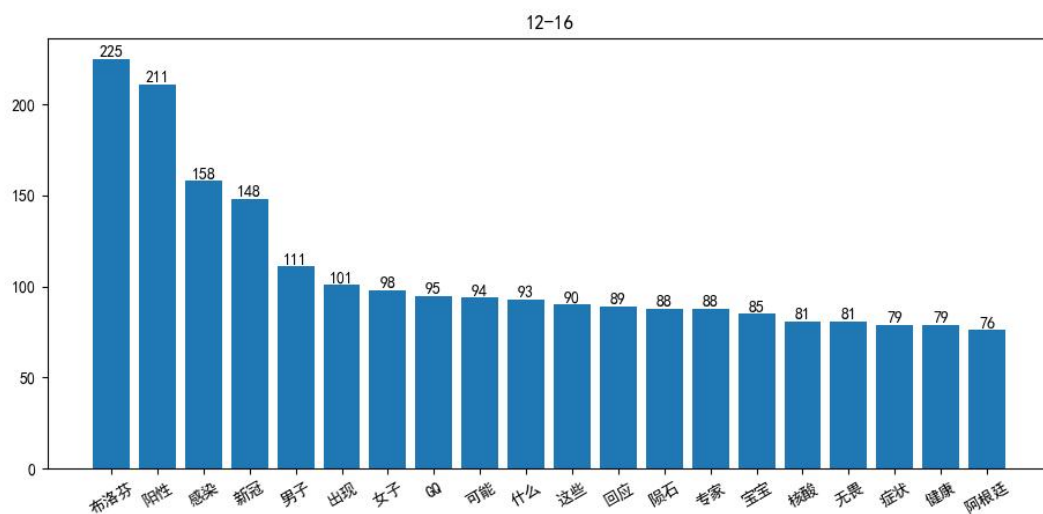
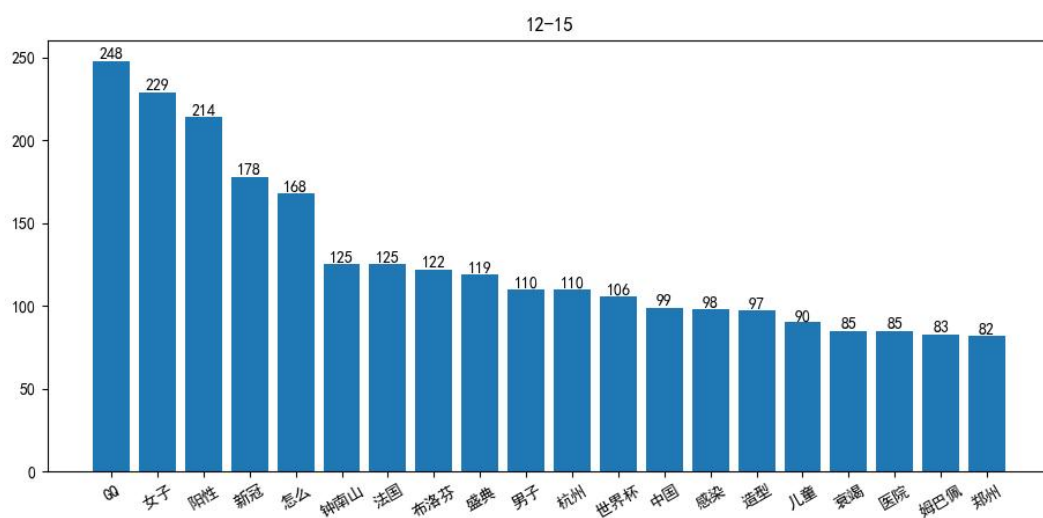
用 matplotlib.pyplot 画图，图表字体设置为中文黑体：

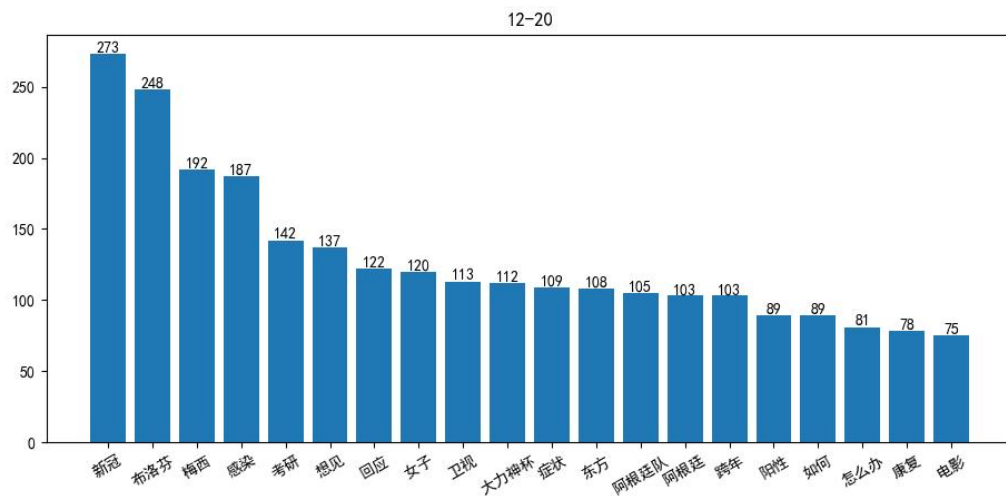
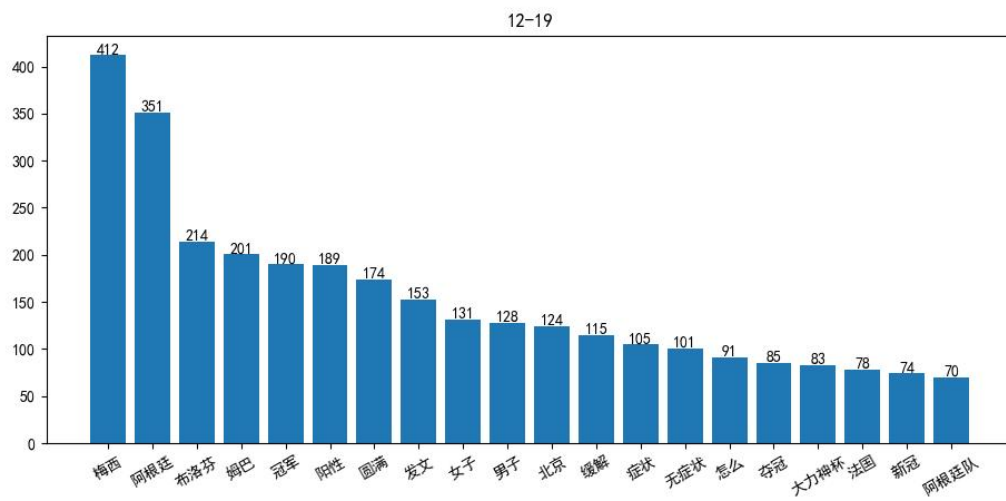
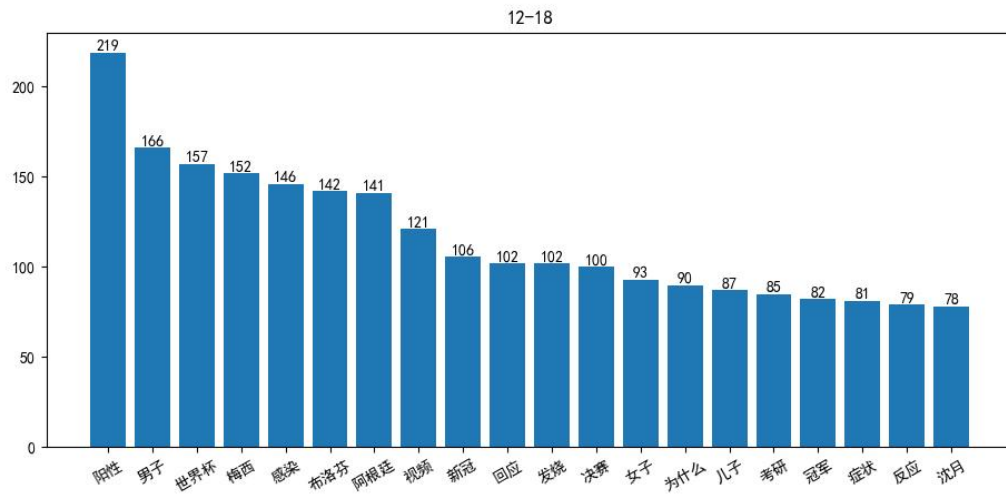
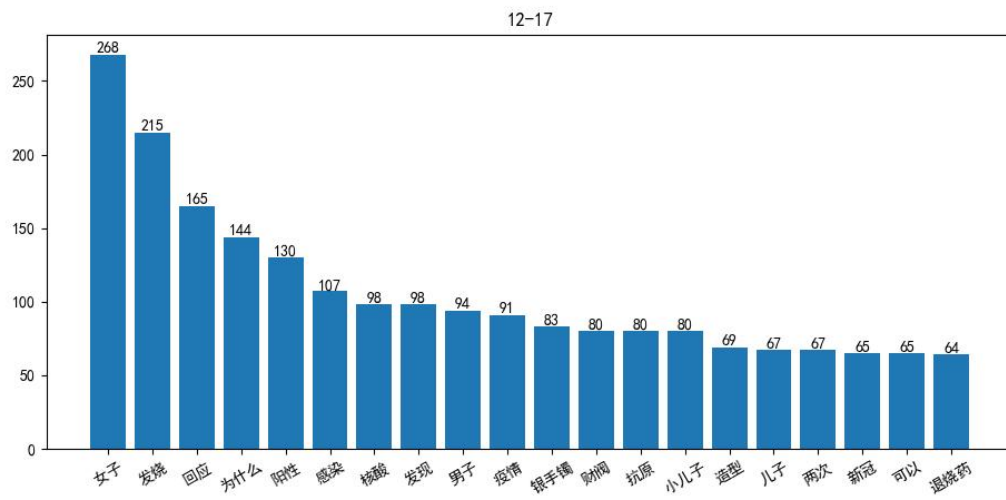
```
1. plt.rcParams['font.sans-serif'] = ['SimHei'] # 中文设置为黑体
```

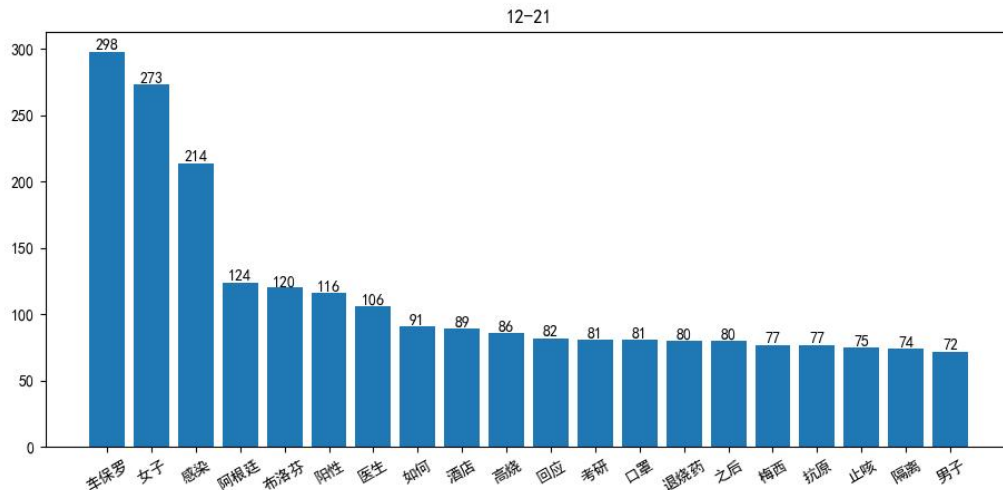
标签太长显示不太友好，旋转 30 度：

```
1. plt.xticks(rotation = 30) # x 轴标签旋转 30 度
```

最终效果：







4.3.3 动态柱状图

这个是用 pyecharts 制作的，也是花费了很多力气，
最后用 jupyter notebook 加了这两行才能网页展示：

1. `from pyecharts.globals import CurrentConfig, NotebookType`
2. `CurrentConfig.NOTEBOOK_TYPE = NotebookType.JUPYTER_NOTEBOOK`

预处理只需要合并 7 个 csv 文件，去除空值行，其他不需要调整，共 29750 条。

```

f_1 = pd.read_csv('D:/code/days/12_15.csv', header = None, encoding = 'utf-8-sig')
f_2 = pd.read_csv('D:/code/days/12_16.csv', header = None, encoding = 'utf-8-sig')
f_3 = pd.read_csv('D:/code/days/12_17.csv', header = None, encoding = 'utf-8-sig')
f_4 = pd.read_csv('D:/code/days/12_18.csv', header = None, encoding = 'utf-8-sig')
f_5 = pd.read_csv('D:/code/days/12_19.csv', header = None, encoding = 'utf-8-sig')
f_6 = pd.read_csv('D:/code/days/12_20.csv', header = None, encoding = 'utf-8-sig')
f_7 = pd.read_csv('D:/code/days/12_21.csv', header = None, encoding = 'utf-8-sig')
l = [f_1, f_2, f_3, f_4, f_5, f_6, f_7]
data = pd.concat(l)
data = data.dropna()
print(data.shape[0])
'''tl = Timeline({"theme": ThemeType.MACARONS})
for i in range(588):
    bar = (Bar()
        .add_xaxis(list(data.iloc[:, 0])[i*50 : i*50 + 20][::-1])
        .add_yaxis("微博热搜榜", list(data.iloc[:, 2])[i*50 : i*50 + 20][::-1].reversal_axis())
        .set_series_opts(label_opts = opts.LabelOpts(position = "inside"))
        .set_global_opts(title_opts=opts.TitleOpts(list(data.iloc[:, 4])[i*50 : i*50 + 1])))
    tl.add(bar, '')
    grid = (Grid().add(bar, grid_opts=opts.GridOpts(pos_left = "20%", pos_right = "0%"))) #将图形整体右移
    tl.add(grid, '')
    tl.add_schema(play_interval = 150, is_timeline_show = True, is_auto_play = False, is_loop_play = False)

tl.render('hot.html')'''

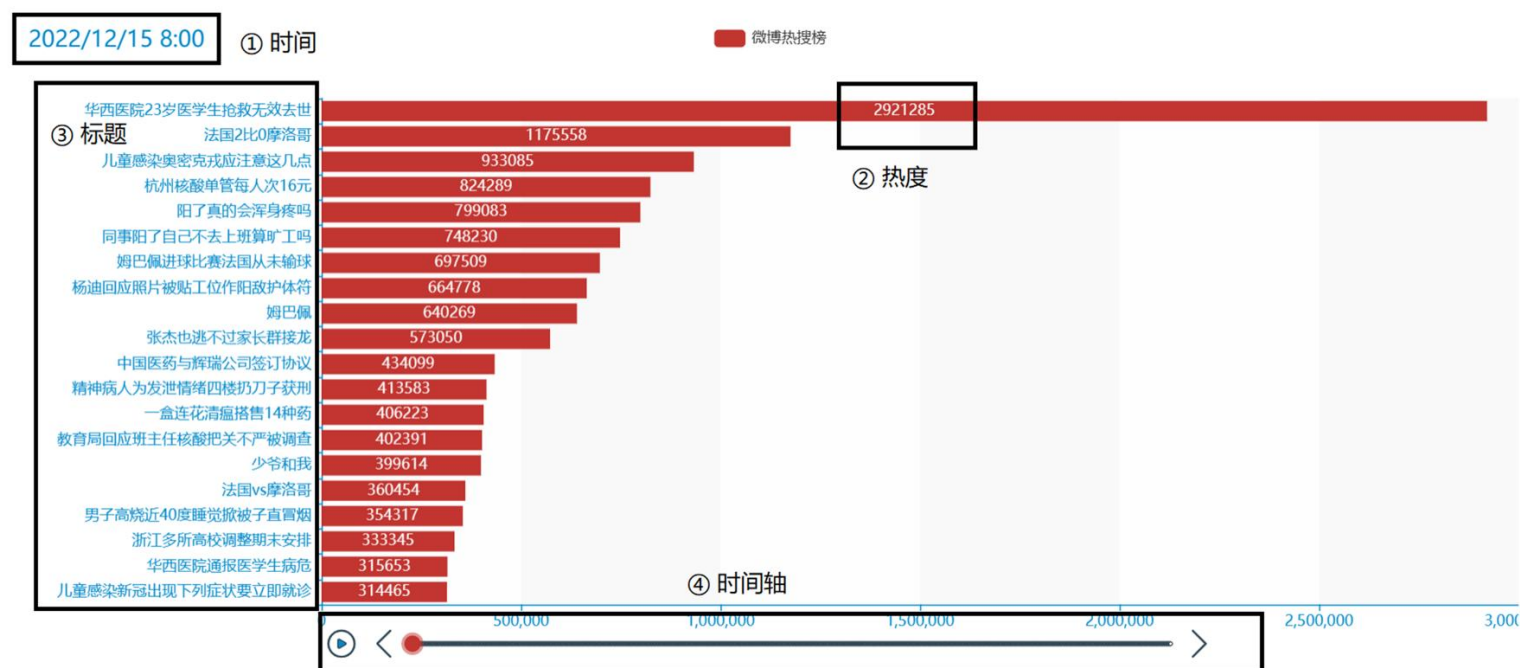
```

①② 由于大小限制以及美观，动态柱状图每次展示前 20 条热搜标题和热度；

③ 标题显示时间，随着展示的时间轴变化。

最后动态展示速度设置为间隔 150ms，便于看清内容。

静态的效果图：

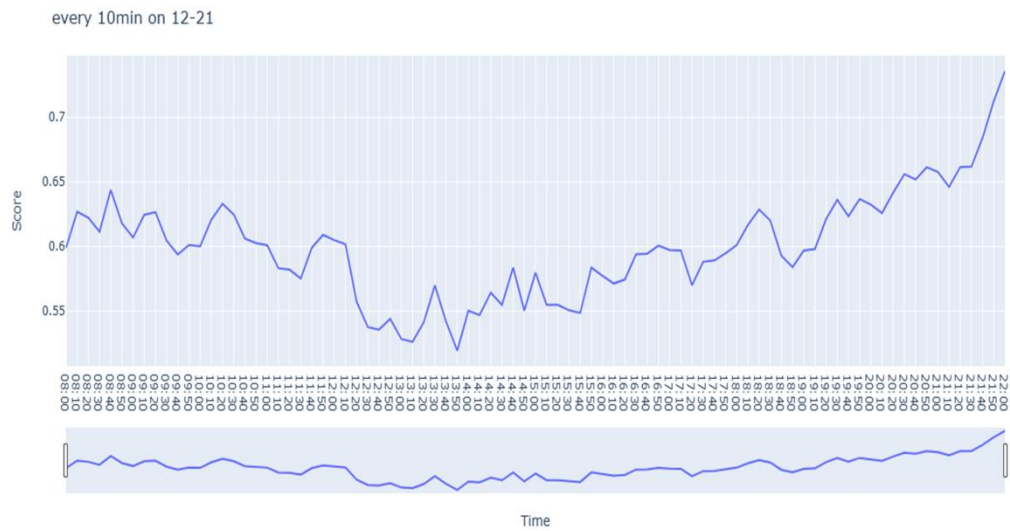


动态效果图可以在附件运行 hot.html 查看，时间轴可以播放也可以随意拖动。

4.3.4 情感分析

情感分析做了三个版本，一天中的情感分析（间隔 10 分钟）、七天的情感分析（间隔 1 天）、七天的情感分析（间隔 1 小时）。用 plotly.express 绘制，x 轴可以放大缩小拖动。

①首先是第一个，以 21 号为例，当天下午得分较低，晚上得分持续走高。总体评分是积极的，其他六天也是如此，看来互联网舆论控评还是往积极走。



② 其次是七天的情感分析均值对比



从中可以很清晰地看出，评分最低的是12月16号，最高是19号，结合先前的词云图以及top20，可以看到16号主要是“布洛芬”“阳性”“感染”“新冠”，而18号23点世界杯决赛开始，19号的关键词是“阿根廷夺冠”“梅西”“布洛芬”等。放开以后疫情对人们的打击还是很沉重的，而世界杯，特别是阿根廷夺冠，对人们精神的鼓舞也很显著。

③ 最后是七天内间隔一小时的情感分析图：



这张图波动就比较直观了，可以清楚地看到每天的波峰波谷，直接根据时间查到对应那一小时的热搜，美中不足的是缺乏晚上的统计，无法比较 emo 到底是晚上多还是白天多，但微博热搜的情感倾向是有所控制的，即使有统计也并不准确，不如去看网易云评论。所以每天只截取到晚上十点。

三张图要注意时间的获取，十分钟是隔 50 条取一次，且一天内的只获取后半几点几分的时间：

```
1. time = []
2. for i in range(0, len(data), 50): # 每隔 50 行取数据
3.     date = data.iloc[i, 4].split()[1]
4.     time.append(date)
```

一天是隔 4250 条取一次，只获取几号这个世界，按空格划分：

```
1. time = []
2. for i in range(0, len(data), 4250):
3.     date = data.iloc[i, 4].split()[0]
4.     time.append(date)
```

一小时是隔 300 条取一次，同样只获取后半时间：

```
1. time = []
2. for i in range(0, len(data), 300):
3.     date = data.iloc[i, 5].split()[1]
4.     time.append(date)
```

clean(f_1)函数处理后重新标号，会多一行行号，所以 data.iloc[i, 5]，不是第 4 列。同时，为了明确七天划分的界限，增加了虚线和文本示意：

```
1. fig.add_trace(go.Scatter(x = [7, 21, 35, 49, 63, 77, 91],
```

```

2.         y = [0.715, 0.715, 0.715, 0.715, 0.715, 0.715, 0.
715], line_dash = 'dash',
3.         text = ['12-15', '12-16', '12-17', '12-18', '12-
19', '12-20', '12-21'],
4.         mode = 'text'))
5.     for i in range(6):
6.         i += 1
7.         fig.add_vline(x = 14*i, line_dash = 'dash')

```

4.3.5 多项式拟合

首先用 dataframe 统计了上榜频次

```

14 print(data.iloc[:, 0].value_counts())

```

PROBLEMS	OUTPUT	TERMINAL	DEBUG CONSOLE
<pre>> python -u "d:\code\INTRODUCTION.py\nihe.py"</pre>			
考研	179		
阿根廷冠军	85		
大概率都会感染越晚变阳症状越轻	71		
姨妈期杀疯了的免疫系统	71		
车保罗	69		

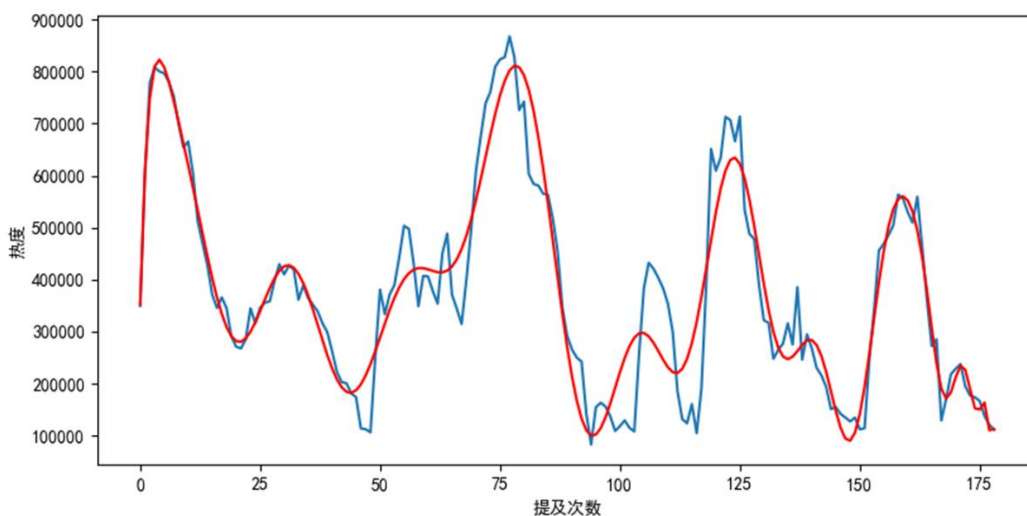
选取了频次最高的“考研”，出现了 179 次，然后用 numpy 库的 polyfit 函数进行多项式拟合，结合 poly1d 函数一起使用。

一开始想用最小二乘法估计参数，但是没找到合适的函数，于是用多项式尝试了一下，拟合阶数为 70 次：

```

1. z1 = np.polyfit(x, y, 70)
2. p1 = np.poly1d(z1)
3. fx = p1(x)    # 求对应 x 的各项拟合函数值
4.
5. # 绘制坐标系散点数据及拟合曲线图
6. plot1 = plt.plot(x, y, label = 'origin data')
7. plot2 = plt.plot(x, fx, 'r', label = 'polyfit data')

```



可以看出，拟合的结果还是比较准确的。提及次数与热度变化没有必然的联系。整体趋势是波动下降的，有点像自由落体，前面波峰大但间隔远，后面波峰多间隔近。

5 实验评价

该项目是基于微博热搜榜的一系列数据可视化与分析，中间有遇到很多困难，不管是技术上还是生活中，但最终还算是圆满落幕。爬虫是遇到的第一个难关，本来是想爬取热搜时光机 (<https://www.weibotop.cn/2.0/>)，可以爬取历史数据，扩大数据集，但是加密太好了，且爬虫只是开头，后续还有很多工作时间不好把握，于是直接爬取微博 (<https://s.weibo.com/top/summary?cate=realtimehot>)，还用任务计划程序反复执行，不愧是万事开头难。项目中的难度还有中文文本处理和画图，比如词语分割，nlp 真的好难，所以最后还是用了 snownlp 库，没有建模。画图也有各种库：plotly.express、plotly.graph_objects、matplotlib.pyplot 和 pyecharts，很多图表类型是第一次画，比如动态柱状图和交互式折线图。最后多项式拟合，本来也打算建模的，还是用了 numpy 库。

总的来说，对于这个多事之秋，项目完成得还可以。

6 后续工作

该项目还有如下一些想法没来得及实现，打算在后续优化完成：

- ① 重新或继续爬虫扩大数据集，以实现更多探索。其实最开始想做全年的情感分析，但是数据集无法支持，就对这七天做了详细分析。
- ② 建立情感分析模型。比如用 TF-IDF 进行文本特征处理，再建模。
- ③ 建立拟合和热搜预测模型。后来想到了 Logistic 逻辑回归模型，可以把热度改为累计热度，初期增长迅速，后期放缓，呈 s 型，再用最小二乘法估计拟合，应该可以。

7 结论

- ① 结合词云和情感分析，得出：未感染人群对感染新冠很担忧，感染人群很痛苦，布洛芬很热销。阿根廷夺冠网友很激动，梅西第一次世界杯夺冠。
- ② 通过每小时的情感分析，得出：除 15 号外，其余每天 8:00 到 9:00 微博热搜的负面情感有所增加，网友都不太喜欢早起。
- ③ 通过动态柱状图，得出：通常热搜第一的热度一骑绝尘，偏向实时爆炸性新闻和社会焦点，热度要高出好几倍，偏向昙花一现。
- ④ 通过多项式拟合，得出：提及次数与热度并不是正相关，只能说明一直有持续关注，提及次数与热度变化没有必然的联系。整体趋势是波动下降的，有点像自由落体，前面波峰大但间隔远，后面波峰多间隔近。更爆炸性的新闻在不断诞生冲击榜首。