

## 整体思路分析报告

整个项目的思路一共分为以下几个步骤：

1. 先去微博获取数据
2. 对数据进行清洗，删除重复项内容
3. 再去对数据进行可视化处理

## 获取数据方法一共分两步走

首先找到网页版微博，然后输入对应的标签，#深圳疫情#，#西安疫情#

然后进行高级搜索

<https://s.weibo.com/weibo?q=%23%E8%A5%BF%E5%AE%89%E7%96%AB%E6%83%85%23>



选择对应的时间，这里有一点是需要注意的，因为显示的缘故，一次最多只能显示50页



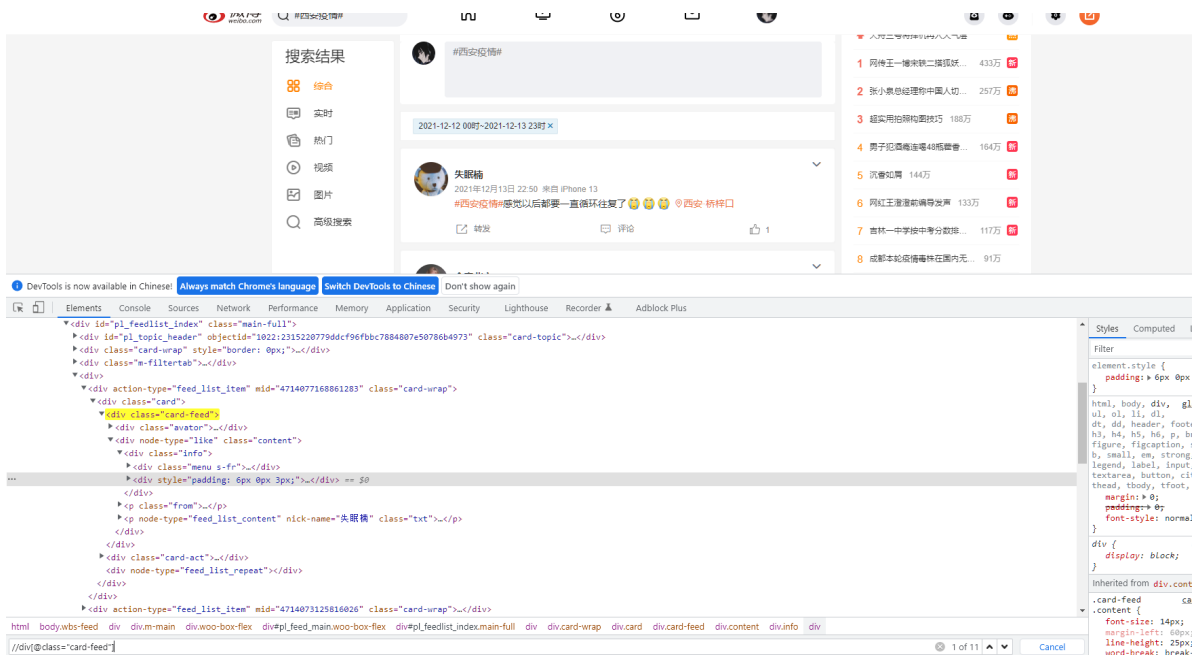
所以程序这边设置了，去获取每一天的内容，每一天都爬取50页，确保信息收集齐全  
 思路如下：

```

rng = pd.date_range(start='2/13/2022', end='4/1/2022')
q = quote('#深圳疫情#')
list_time = []
for r in rng:
    r = str(r).split(" ")
    list_time.append(r[0])
for l in tqdm(range(len(list_time)-1)):
    for i in range(1,51,1):
        url = 'https://s.weibo.com/weibo?q={}&typeall=1&suball=1&timescope=custom: {}: {}&Refer=g&page={}'.format(q,list_time[l],list_time[l+1],i)
        main_ur(url)
  
```

然后根据这个时间段去爬取

这里采取的是xpath定位法



首先先定位该页面的一整个文本内容，获取每一页每一条正文发布的内容，不同页正文数量不同，所以这样自动化去定位到每一页具体的正文内容比较便利

这一整块内容就是内容定位思路

获取好全部内容之后，保存为CSV文件

接着我们再去获取官号下面的推文

<https://weibo.cn/3757167087/profile?keyword=%E7%96%AB%E6%83%85%E9%80%9A%E6%8A%A5&hasori=0&haspic=0&starttime=20211212&endtime=20211230&advancedfilter=1&page=2>

[illegible]

微博这边是很多个网站的，数据都是互通的，只是每个网站的定位不同，这个网站是为开发者服务的  
然后我们进行筛选和输入关键词



The screenshot shows the Weibo search interface for a specific user profile. At the top, there's a navigation bar with '首页' (Home), '消息' (Messages), and '刷新' (Refresh). Below it, a link '手机微博触屏版, 点击前往>>' (Mobile Weibo touch screen version, click to go to >>) is visible. The main search area has a '返回西安发布的微博' (Return to Weibo posted in Xi'an) link. The search criteria are: '关键字: 疫情通报' (Keyword: Epidemic Report) with a '(必填)' (Required) label. There are radio buttons for '是否原创: 全部' (Originality: All), '原创' (Original), and '转发' (Retweet). Another set of radio buttons for '是否带图: 全部' (With image: All), '有图' (With image), and '无图' (Without image). The '发布时间: 20211212' (Release time: 20211212) to '20211230' (20211230) range is set. A '筛选' (Filter) button is present. Below the search criteria, there's another '返回西安发布的微博' link. At the bottom, there's a navigation bar with '首页' (Home), '微博优选商城' (Weibo Preferred Mall), '反馈' (Feedback), '客户端' (Client), '举报' (Report), and '退出' (Exit). A '设置: 图片, 条数, 隐私' (Settings: Image, Number of items, Privacy) link is also visible.

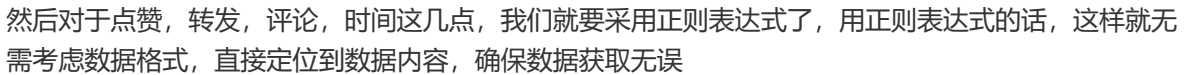
该网页每次最多只能返回100页的数据，根据这个逻辑

```
rng = pd.date_range(start='02/13/2022', end='04/01/2022', freq='w')
list_time = []
for r in rng:
    r = str(r).split(" ")
    list_time.append(r[0].replace('-', ''))
list_time.append('20220401')
str_list = ['疫情通报', '清零', '疫情防控', '密接', '新闻发布会']
for s in str_list:
    q = quote(s)
    for l in tqdm(range(len(list_time)-1)):
        for i in range(1, 101, 1):
            url = 'https://weibo.cn/2831150640/profile?keyword={}&hasori=0&haspic=0&starttime={}&endtime={}&advancedfilter=1&page={}'.format(q, list_time[l], list_time[l+1], i)
            main_ur(url)
```

这样确保可以获取全部内容

在数据定位上，该网站会分两种情况，为了把全部数据获取到，我们这边要换一种思路

首先对于内容定位，我们还是采用xpath语法，在内容方面都是统一的，来获取全部的内容



## 数据清洗

[illegible]

从这些数据上来看，首先乱，第二就是可能会有重复性内容，所以这个清洗工作还是刚需来的

数据清洗一共分8步骤

首先清洗时间列的内容

```
def main1(x):
    x1 = str(x)
    x1 = x1.replace("'", "").replace("[", "").replace("]", "").replace("
", "").replace("\n", "")
    x1 = str(x1)
    x2 = x1.split('\n')
    return x2[1]
```

把一些无意义的符号和回车全部删除

第二步便是清洗博主列

```
def main2(x):
    x1 = str(x)
    x1 = x1.replace("'", "").replace("[", "").replace("]", "").replace("
", "").replace("\n", "")
    x1 = str(x1)
    x2 = x1.split('\n')
    return x2[0]
```

同样是把无意义的符号和回车全部删除

然后其他的列同样如上

```
df['时间'] = df['时间'].apply(main1)
df['博主'] = df['博主'].apply(main2)
df['认证'] = df['认证'].apply(main3)
df['内容'] = df['内容'].apply(main4)
df['点赞'] = df['点赞'].apply(main5)
df['转发'] = df['转发'].apply(main6)
df['评论'] = df['评论'].apply(main7)
df = df.drop_duplicates(subset=['内容'], keep='first')
print(df)
```

把一些无效的内容全部删除，保留有效数据

做好上面的工作之后，然后对内容这一列进行去重工作，把重复内容项全部删除

```
df = df.drop_duplicates(subset=['内容'], keep='first')
```





接着再对时间进行筛选

保留符合时间段的内容，时间段为2020年12月到2021年1月的全部内容，其他时间全部删除,另一个城市做法如是，只是筛选时间段的写法不同而已，思路一致

```
def main8(x):
    x1 = str(x)
    if '01月' in x1:
        x1 = '2022年' + x1
    if '12月' in x1:
        return x1
    elif '01月' in x1:
        return x1
    else:
        return np.NaN
```

把数据全部清洗好过后，剩下就是删除不符合逻辑的数据，和时间以及重复项内容，保存为一个新的文件

然后我们再去看看新的文件大小如何，对比之前的数据，足足少了一半，

 new_深圳疫情.csv	2022/7/18 9:55	Microsoft Excel ...	4,572 KB
 new_西安疫情.csv	2022/7/18 10:03	Microsoft Excel ...	5,469 KB
...			
 #深圳疫情#.csv	2022/7/15 22:44	Microsoft Excel ...	12,257 KB
 #西安疫情#.csv	2022/7/15 21:15	Microsoft Excel ...	11,883 KB

新的文件如上，数据干净了不少，然后去完重之后，只剩2万多条数据，比之前3万多条数据（两个城市的数据量都是3万起步）

说明重复内容还是很多的

## 最后便是数据分析

数据分析这边，一共分两个城市去分别对其进行分析，其思路大同小异

```
import pandas as pd
import matplotlib.pyplot as plt

df1 = pd.read_csv('new_深圳疫情.csv')
df1 = df1.drop(['博主', '认证'], axis=1)
df2 = pd.read_csv('深圳微博发布厅.csv')
df3 = pd.read_csv('深圳卫健委.csv')

data = pd.concat([df1, df2, df3], axis=0)

data['数量'] = 1
```

首先读取和深圳相关的全部文件，然后把这些文件全部合并起来，整成一个新的总的文件

```
def time1(x):
    x = str(x)
    x1 = str(x).split('日')
    x1 = '2022年' + x1[0] + '日'
    return x1
```

然后重新整合一下它的时间排序，全部统一一下，然后把时间作为序列，用于后面数据可视化

```
data['时间'] = data['时间'].apply(time1)
data['时间'] = pd.to_datetime(data['时间'], format='%Y年%m月%d日')
data.index = data['时间']
```

#时间趋势

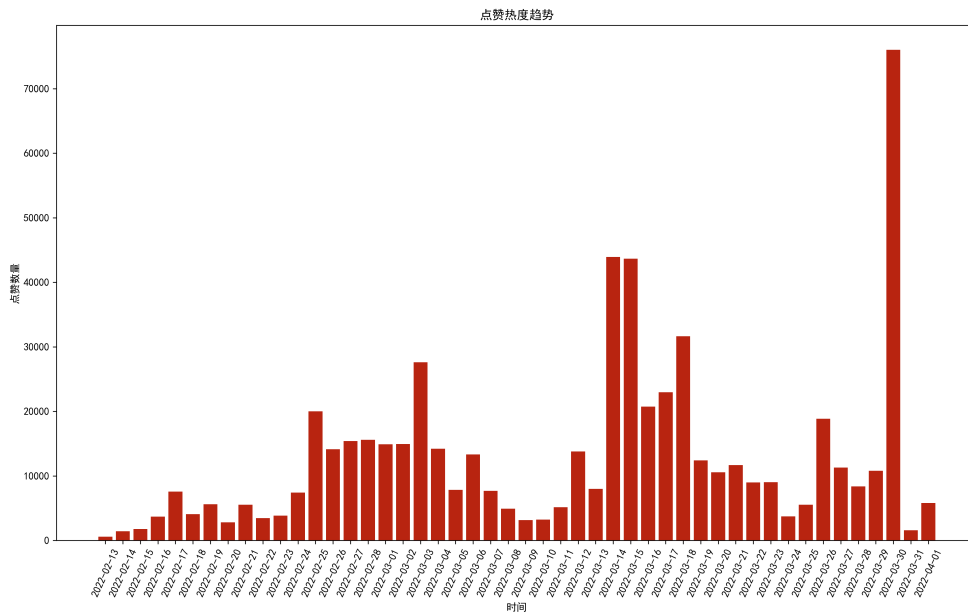
```
def main1():
    new_df = data['数量'].resample('D').sum()
    x_data = [str(n).split(" ")[0] for n in new_df.index]
    y_data = list(new_df.values)
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.figure(figsize=(16, 9), dpi=300)
    plt.plot(x_data, y_data, linewidth=3, color='#EC7063')
    plt.title("时间热度趋势")
    plt.xlabel("时间")
    plt.ylabel("发帖数量")
    plt.xticks(rotation=65)
    plt.savefig('./深圳数据可视化/发帖热度时间趋势图.png')
    plt.show()
```

然后把数据弄成天的趋势，再去读取每天发帖的数量，然后再去用折线的方式表达出来



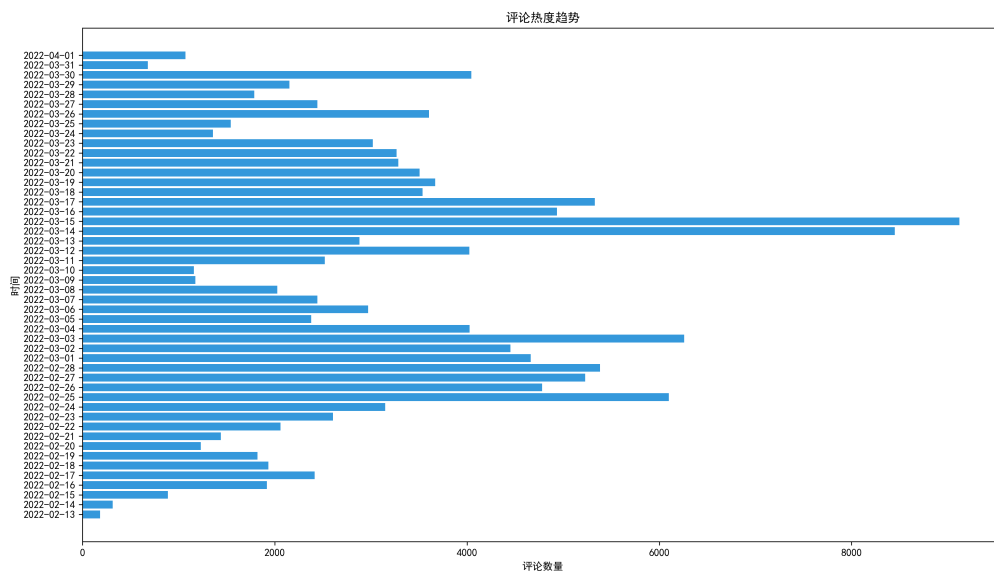
从该图中可以看出，在2月底以及3月中旬是发帖的高峰期，这时候发帖的数量最多

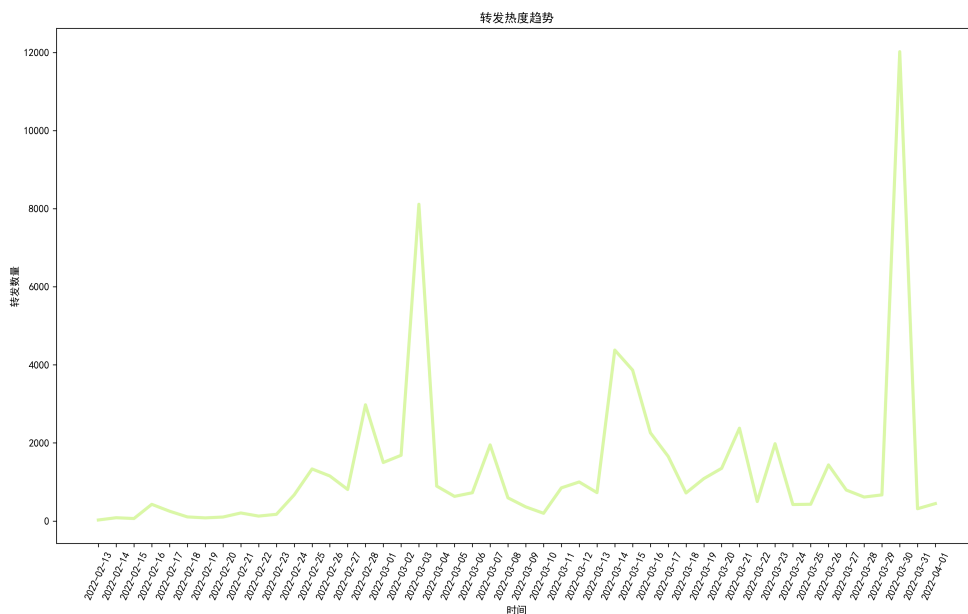




而点赞最多的则是在3月30号，这时候应该是某一条评论，说中了人们的点，使得点赞量飙升

而讨论量最多的则是在15号这天，这个和发帖量有关，发帖量多，评论多，说明这一天是发生了一些重大事件，要深入内容进行探讨了





转发趋势则是和点赞量呈正相关，应该就是某一条正文的出现，使得点赞剧增，从而带动人们去转发这条推文，使得这两个指标指数飙升