基于社交媒体的突发公共卫生事件的公众情绪研究

—— 计算社会学 - 情绪分析与预测

一. 研究背景与研究问题

当下新型冠状病毒（COVID-19） 肆虐全球，给人们的生产和生活产生了极大影响，也形成了疫情下独特的网络社会心态和公众情绪。 借助社交媒体，公众能够快速交换风险信息， 提高对危机形势的认识、减少公众的恐慌，并在疾病大流行期间建立公众的信任，为危机应对做好准备。危机期间，利用社交媒体，监控公众对未知疾病的认识和理解对于危机管理至关重要。

本次作业的研究问题，立足此次新型冠状病毒 （ COVID-19）传播的这一重大公共卫生事件，基于社交媒体的数据，运用机器学习方法，进行大众网络社会情绪分析和预测。主要目的是分析出不同阶段公众在社交媒体上所展现出的主要情绪、次要情绪与情绪变化。借助机器学习与数据分析的手段，准确并客观地了解公众的网络情绪与基于此呈现出的规律，更有利于舆论的引导与控制，同时引导大众以更加积极的心态应对处理公共卫生事件及其衍生问题。本作业数据来源为新浪微博。

二. 研究思路与方法

2.0 主要思路

0. 本作业通过**微博的热点新闻和高赞评论**来追踪公众在社交媒体上的情绪。
1. 首先利用python**爬虫**爬取了主要新闻媒体疫情期间与疫情相关的新闻（包括时间正文评论以及点赞评论转发数量）共13718条。
2. 紧接着**筛选**出重点新闻（点赞评论转发数量高于平均值）的评论作为机器学习的训练集
3. 人工对这些评论**标注情绪分类**，归入到不同类目**建立语料库**。
4. 之后运用**有监督学习**对语料库进行训练。
5. **训练**出的模型就可以对疫情的**四个阶段**的新闻评论文本进行**情绪分类**，从而进行情绪的**统计分析**，找到**各阶段公众在社交媒体上所展现出的主要情绪、次要情绪与情绪变化。**

2.1 数据获取

数据集：

新浪微博自2019-12 至 2020-06之间，新浪新闻、央视新闻、人民日报、凤凰新闻四个媒体，与疫情相关的新闻数据，每一项数据基本格式与内容如下：

```
  {
    "url": "https://weibo.com/2803301701/IniAErTYL",
    "content": "……",
    "time": "2019-12-31",
    "like_num": 133000,
    "comment_num": 6262,
    "forward_num": 7106,
    "comments": [ "……",
                  "不传谣不信谣，希望平安无事🙏",
                  "2019最后一天 武汉人民平安",
                  "天佑武汉，天佑中华", "希望是虚惊一场，愿平安！" ]
  }
```

数据获取方法：

大规模采集数据使用爬虫来完成,而微博的内容都是动态呈现的,需要将页面不断下拉才能加载出来。因此选择`selenium`库来完成自动化数据获取.

整体的过程为:**按页发起`get`请求->向下滚动至加载完全->查找并获得相应微博的信息**.

页面分析

新浪微博 PC 端必须要登录才会加载内容,比起手动输入账号密码,使用 cookie 更加方便快捷:

在对新浪微博发起`get`请求,手动登录后保存 cookie:

```
url = https://weibo.com/sinapapers?is_all=1&stat_date=202012&page=1#feedtop
wd = webdriver.Chrome("chromedriver.exe")
wd.get(url)
time.sleep(10)  # 手动登录
cookies = wd.get_cookie()
with open("cookies.json","w") as f:
    f.write(json.dumps(cookies, ensure_ascii=False))
```

之后在发起`get`请求时,添加 cookie,即可完成自动登录:

```
def add_cookies(wd):
    with open("cookies.json", 'r') as file:
        cookie_list = json.loads(file.read())

    wd.delete_all_cookies()
    for cookie in cookie_list:
        wd.add_cookie(cookie)

wd.get(url)
add_cookies(wd)
wd.get(url)
```

以 "新浪新闻" 为例,PC 版网页端的 URL 为 `https://weibo.com/sinapapers?is_all=1&stat_date=202012&page=1#feedtop`,其中 `stat_date` 参数确定了微博的时间（年/月）, `page` 确定了当前页数,因此以它为基础调整参数,发起 `get` 请求,再从这页中获取相关的数据:

```
base = r"https://weibo.com/{0}?is_all=1&stat_date={1}{2}&page={3}#feedtop"
for year in range(2019, 2021):
    for month in range(1, 13):
        if year == 2019 and month < 12:
            continue
        if year == 2020 and month >= 7:
            break

        pages = get_pages_num(wd, url)

        for page in range(0, pages)::
            url = base.format(str(year),str(month).zfill(2),page+1)
            data_lsit.append(get_data_from_a_page(wd, url))
```

元素获取

判断页面加载完全需要检查下图翻页框的存在, 若存在则停止向下滚动.

```
def scroll_down(wd):
    attemps = 0
    while len(wd.find_elements_by_xpath("//div[contains(@node-type,'feed_list_page')]")) == 0:
        wd.execute_script("window.scrollTo(0,document.body.scrollHeight)") # 滚动至页面底部
        attemps += 1
        if attemps == 6:
            wd.refresh()     # 超过最大尝试次数则刷新
            attemps = 0
        time.sleep(1)
```

元素的获取只需要通过 Chrome 的开发者工具定位到位置,此处选择 `find_elements_by_xpath()` 这个方法,下面是获取翻页框元素的例子:

```
wd.find_elements_by_xpath("//div[contains(@node-type,'feed_list_page')]")
```

这个方法会返回所有符合搜索条件的元素组成的列表, 若没找到则返回空列表. 因为网页加载需要一定的时间, 此处采用**显式等待**, 即指定一个最长等待时间和等待条件, 在等待时间内判断条件是否满足, 若满足则返回, 否则抛出异常, 以下是获取一页中所有微博内容元素的代码:

```
elements = WebDriverWait(wd, timeout=5, poll_frequency=0.5).until(
                lambda x: wd.find_elements_by_xpath("//div[contains(@action-data,'cur_visible=0')]"))
```

在此处经过一次初步的筛选, 若内容中没有与 "疫情", "新冠" 等相关的内容则舍弃.

到这一步, 任务就只剩下点击评论按钮然后获得评论, 但是实际测试时发现, 微博的网页版很多时候并不能顺利地加载出评论, 使得最终获得的评论数据都是空的, 而移动版的微博加载出评论的成功率要高得多, 除非评论区被关闭或者强制要求客户端打开.

所以思路就转变为: **PC 版网页获得微博的链接->转换为可以由移动版访问的链接->在移动版页面获取相关信息.**

经查阅资料, 发现转发框中可以获取到供移动端直接访问的链接:

调用 `find_elements_by_xpath()` 找到转发和私信的按钮, 然后调用 `element.click()` 方法点击, 获取链接, 最后关闭转发微博框.


```
button = WebDriverWait(wd, timeout=5, poll_frequency=0.5).until(
                lambda x:
wd.find_elements_by_xpath("//div[contains(@action-data,'cur_visible=0')]["
+ str(
                    index + 1) + "]//ul[@class='WB_row_line WB_row_r4
clearfix S_line2']/li[2]/a"))
wd.execute_script('arguments[0].click();', button[0]) # 点击

forward_button = WebDriverWait(wd, timeout=5, poll_frequency=0.5).until(
                lambda x: wd.find_elements_by_xpath('//div[@class="W_layer
"]//li[3]'))
forward_button[0].click()   # 另一种点击

mobile_url = WebDriverWait(wd, timeout=5, poll_frequency=0.5).until(
                lambda x: wd.find_elements_by_xpath(
                    '//div[@node-
type="toMessage_client"]//div[@class="WB_feed_publish
clearfix"]//div[@class="sendbox_area S_bg2"]'))
ActionChains(wd).send_keys(Keys.ESCAPE).perform()   # 按下 ESC 键
```

要使用移动端的 UA 也是很简便的, 只需通过添加 `options` 参数即可:

```
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument(
        'User-Agent=Mozilla/5.0 (Linux; U; Android 8.1.0; zh-cn; BLA-AL00 Build/HUAWEIBLA-AL00) AppleWebKit/537.36 ('
        'KHTML, like Gecko) Version/4.0 Chrome/57.0.2987.132 MQQBrowser/8.9 Mobile Safari/537.36') # 伪装成手机
wd = webdriver.Chrome(r"D:\project\chromedriver.exe", options=chrome_options)
```

最后再用上方提到的元素获取的方法在移动版页面获取到点赞数, 评论数, 转发数以及评论内容即可.

异常处理:

在实践中会出现各种各样的意外, 比如超时, 或者访问过于频繁导致被拦截, 若没有异常处理, 爬取到的数据很容易前功尽弃. 这次爬虫的异常来源及处理方法有:

```
1. selenium 库异常`TimeoutException`: 元素在最大等待时间内未加载完成. 处理方法: 刷新页面 (可调用 `wd.refresh()`), 若超过最大尝试次数则暂停提示检查网络或是否处于被拦截状态.

2. 自定义异常 `NoCommentsException` : 评论数不为零但无法获取到评论, 可能是被删评. 处理方法: 跳过.
```

3. 自定义异常 `ClientRequiredException`: 出于未知原因, 用上述方法获得该连接后, 强制要求在客户端访问. 处理方法: 由于这种情况并非很常见, 同样也选择跳过.
4. 自定义异常 `InterceptedException`: 检测到被拦截. 处理方法: 暂停, 可手动输入任意内容继续尝试. 实际测试时, 只要更换 IP 即可解除被拦截的状态.

2.2 新闻筛选

-

- **新闻筛选方法:**

  由于在爬虫时已经初步筛选，获得的是疫情相关新闻的前五高赞评论，因此本次筛选只是筛选出**点赞评论转发数量之和高于中位数**的新闻作为重点新闻，**这些重点新闻的前五高赞评论集合起来作为训练集**(语料库)。

  - 具体实现时，由于高于中位数的评论达到了50%的数据量，我们随机取了其中约8000条评论作为训练集，之所以没有选取四分位点比如前25%数据是因为需要考虑少数点赞的评论的情绪。前50% + 随机则平衡了重点性与广泛性。

  ```
  data_frame['power'] =
      data_frame['like_num'] + data_frame['comment_num'] + data_frame['forward_num']
  # 依据权值power = 点赞数 + 转发数 + 评论数，找到power的中位数
  power_median = data_frame['power'].median()
  # 将高于中位数的作为训练集
  focus_news = data_frame[data_frame['power'] > power_median].copy()
  ```

2.3 建立语料库

- **建立语料库方法**

0. 主要使用`pandas`库
1. 将上一步新闻筛选获得的**重点评论**的`pandas.DataFrame`作为`lable.putLabel()`输入
2. 遍历**评论**的`pandas.DataFrame`，在控制台打印新闻正文与评论正文，**等待用户输入**该评论对应的情绪
3. 将用户输入的情绪分类保存到该`pandas.DataFrame`中
4. `lable.textClassify()`遍历已经完成标注的评论的`pandas.DataFrame`，**根据不同情绪分类的路径创建评论的文本文件，并将文本文件放置在对应类目文件夹之下**

- **语料库的情绪分类**：**中性、乐观、喜悦、嘲讽、忧虑、愤怒**
  - 中性：无明显情绪比较冷静客观，或者正面情绪与负面情绪均有且比较均衡
  - 乐观：对现在事态表示positive的祝愿、鼓励等乐观情感；对人物的致敬、祝愿；对未来的祈祷、希望；b 包括被当下的事情所感动的情绪；
  - 喜悦：对令人喜闻乐见的结果而表达出的喜悦的情感；对自己祖国和政府的自豪感骄傲感；
  - 嘲讽：对部分事件/行为感到可笑，从而批判/抨击、或者单纯取乐阴阳怪气
  - 忧虑：感到焦虑、烦恼、担心、疑惑等负面情绪，包括伤心难过等负面情绪，也包括对人物的悼念
  - 愤怒：对发生的事件感到生气、愤怒

2.3 训练模型-情感预测

- 主要使用`HanLP`库的python接口`pyhanlp`，利用**朴素贝叶斯分类器**，本质上完成了文本分类的任务
- **机器学习训练方法**：
  - 将上一步获得的情绪分类语料库`data/sentimentCorpus`作为输入
  - 建立**朴素贝叶斯分类器**`NaiveBayesClassifier()`对象，调用`train()`方法，由语料库获得持久的分类模型
    - 先对语料库的每个文件进行特征提取，获得特征向量后，利用朴素贝叶斯法（基于贝叶斯定理将联合概率从转化为条件概率，利用特征条件独立假设简化条件概率计算）来进行朴素贝叶斯分类
  - 可以将新的文本作为输入，调用分类器的`classify()`方法，即可返回概率最大的情绪分类类别，即情绪预测

```
import os

from pyhanlp import SafeJClass

NaiveBayesClassifier =
SafeJClass('com.hankcs.hanlp.classification.classifiers.NaiveBayesClass
ifier')

IOUtil = SafeJClass('com.hankcs.hanlp.corpus.io.IOUtil')


def train(corpus_path):
    model_path = corpus_path + '.ser'
    if os.path.isfile(model_path):
        return NaiveBayesClassifier(IOUtil.readObjectFrom(model_path))
    classifier = NaiveBayesClassifier()
    classifier.train(corpus_path)
    model = classifier.getModel()
```

```
    IOUtil.saveObjectTo(model, model_path)
    return NaiveBayesClassifier(model)


def predict(classifier, text):
    print("《{}》\t属于情绪分类\t【{}】".format(text,
classifier.classify(text)))
```

2.4 划分阶段与数据可视化

新闻四个阶段的分类

- **分阶段的方法**:
-
  - 主要利用`pandas`库和`datetime`库
  - 首先每条新闻的`time`转化为`datetime`格式
  - 然后遍历新闻的`DataFrame`，利用每一条新闻的时间确定其在哪个阶段的时间区间内，筛选出对应阶段的新闻的`Dataframe`，再分别写入各个`json`文件
- ```
  first_start = datetime(2019, 12, 8)  # 2019.12.8发现首例肺炎患者
  second_start = datetime(2020, 1, 22)  # 2020.1.23武汉宣布“封城”
  third_start = datetime(2020, 2, 10)  # 2020.2.10十九省对口支援湖北除武汉外16个市州及县级市
  fourth_start = datetime(2020, 3, 10)  # 2020.3.10各省开始有序复工
  end = datetime(2020, 6, 30)
  ```
- ```
      data_frame['time'] = pd.to_datetime(data_frame['time'])
      first_stage_news = data_frame[((data_frame['time'] >= first_start) & (data_frame['time'] < second_start))]
      second_stage_news = data_frame[((data_frame['time'] >= second_start) & (data_frame['time'] < third_start))]
      third_stage_news = data_frame[((data_frame['time'] >= third_start) & (data_frame['time'] < fourth_start))]
      fourth_stage_news = data_frame[((data_frame['time'] >= fourth_start) & (data_frame['time'] < end))]

      with open(stage_1_path + file_name + '1.json', 'w', encoding='utf-8') as f:
          first_stage_news.to_json(f, orient='records', force_ascii=False)
      ……
  ```

统计分析方法

- 利用已经训练好的情绪分类模型，调用分类器的 `classifier.classify('')`方
- 法，**遍历** `data/stageData/`，为**每一阶段的每一个评论预测其情绪**，并统计**每个阶段**的
  - 每种情绪的评论数
  - **每种情绪的所占比例**（该比例数据用于进一步分析，保存在`src/stageAnalyse/count.py`中
- 利用`matplotlib`库进行可视化

- 先对每个阶段进行每种情绪的比例分析，绘制饼状图
- 再对不同阶段的情绪变化分析，绘制折线图

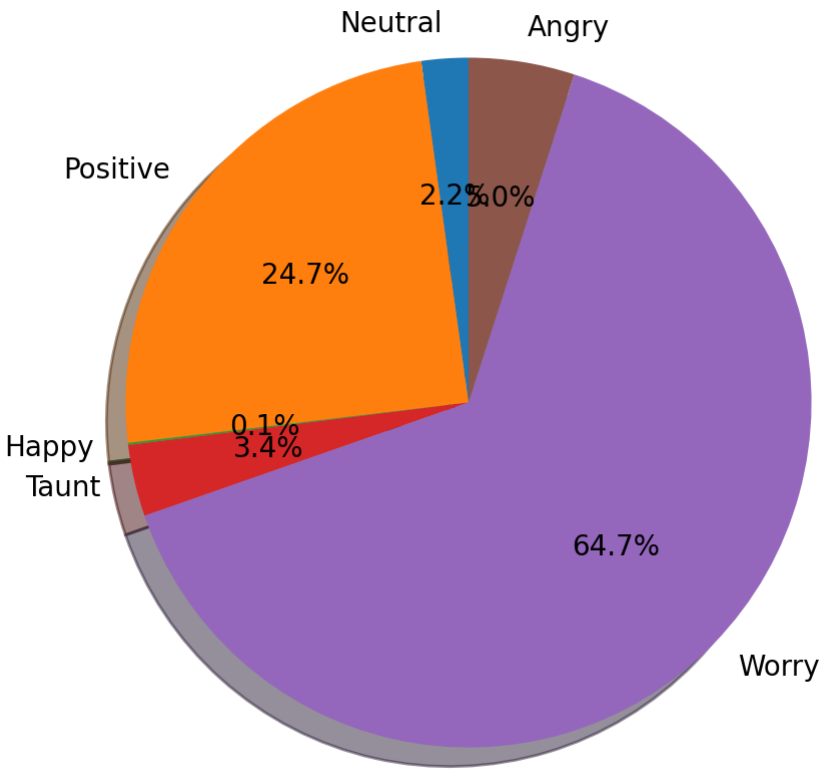
三. 案例可视化分析

3.0 各个阶段划分：

1. 第一阶段：2019.12.8 — 2020.1.22 【2020.12.8 发现首例肺炎患者】
2. 第二阶段：2020.1.23 — 2020.2.9 【2020.1.23 武汉宣布“封城”】
3. 第三阶段：2020.2.10 — 2020.3.9 【2020.2.10 十九省对口支援湖北除武汉外16个市州及县级市】
4. 第四阶段：2020.3.10 — 2020.6.30 【2020.3.10 各省开始有序复工】

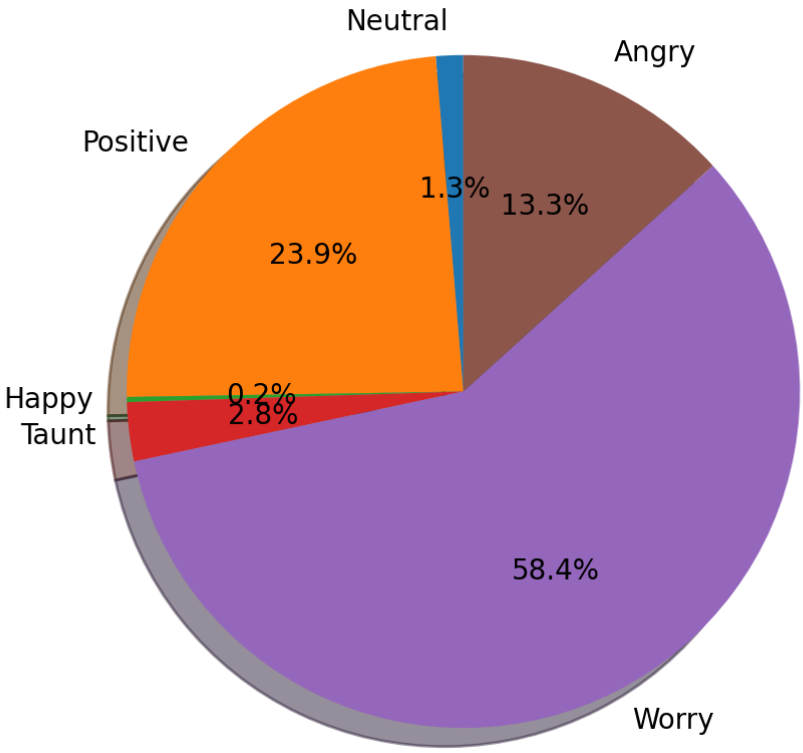
3.1 各个阶段的情绪比例

1. 第一阶段：
   - 中性评论： 23 比例：2.2%
   - 乐观评论：257 比例：24.7%
   - 喜悦评论：1 比例：0.1%
   - 嘲讽评论：35 比例：3.4%
   - 忧虑评论：674 比例：64.7%
   - 愤怒评论：52 比例：5.0%



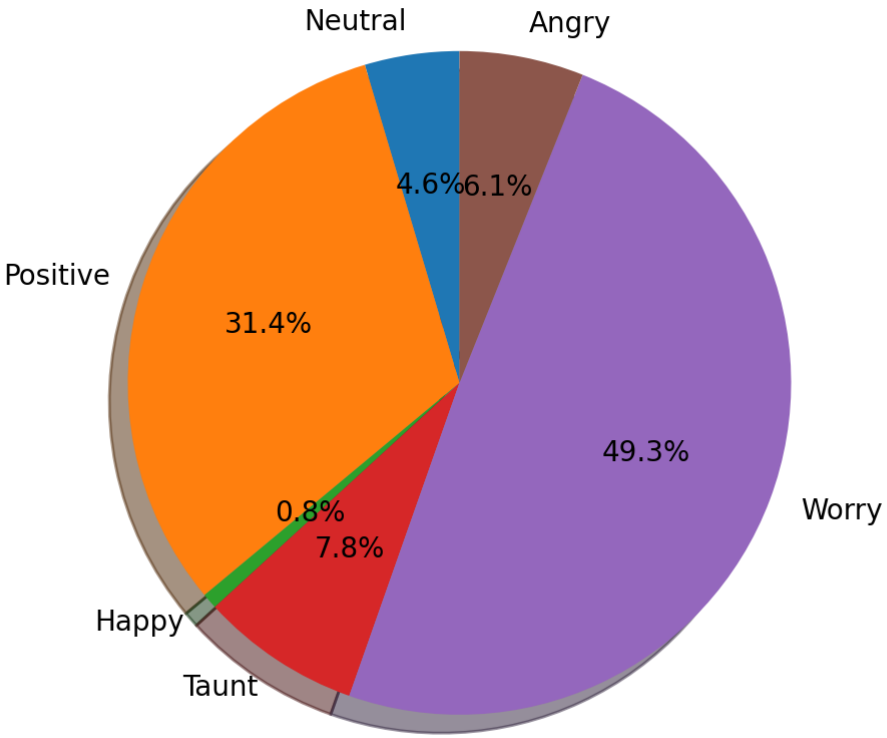
2. 第二阶段：
   - 中性评论： 201 比例：1.3%
   - 乐观评论：3672 比例：23.9%
   - 喜悦评论：38 比例：0.2%
   - 嘲讽评论：434 比例：2.8%
   - 忧虑评论：8950 比例：58.4%

- 愤怒评论：2038　比例：13.3%



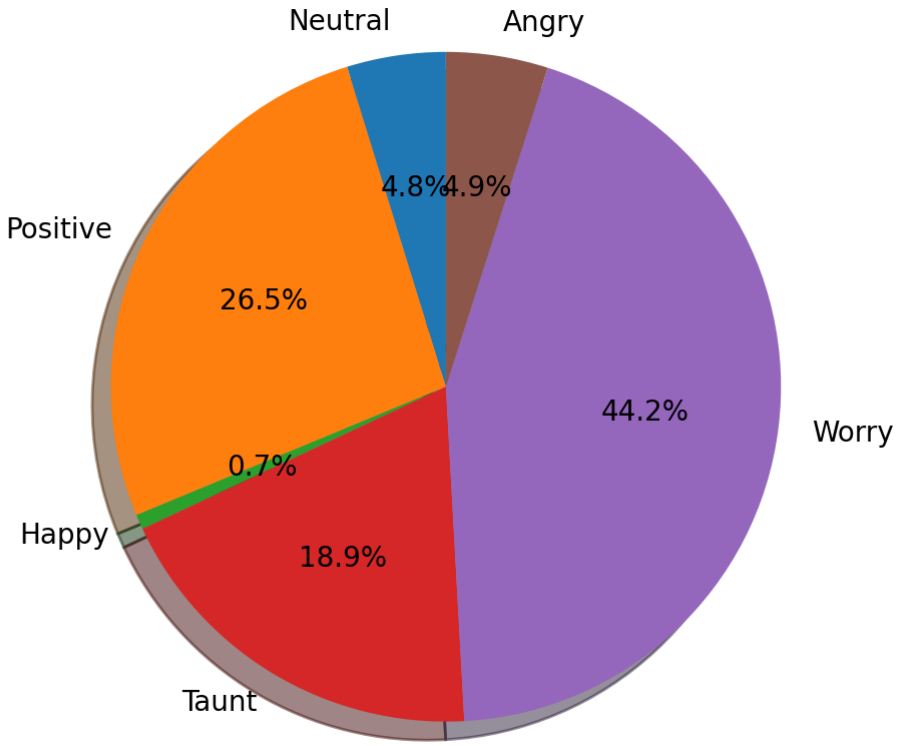
3. 第三阶段：

- 中性评论：　816　比例：4.6%
- 乐观评论：5566　比例：31.4%
- 喜悦评论：134　比例：0.8%
- 嘲讽评论：1382　比例：7.8%
- 忧虑评论：8735　比例：49.3%
- 愤怒评论：1073　比例：6.1%

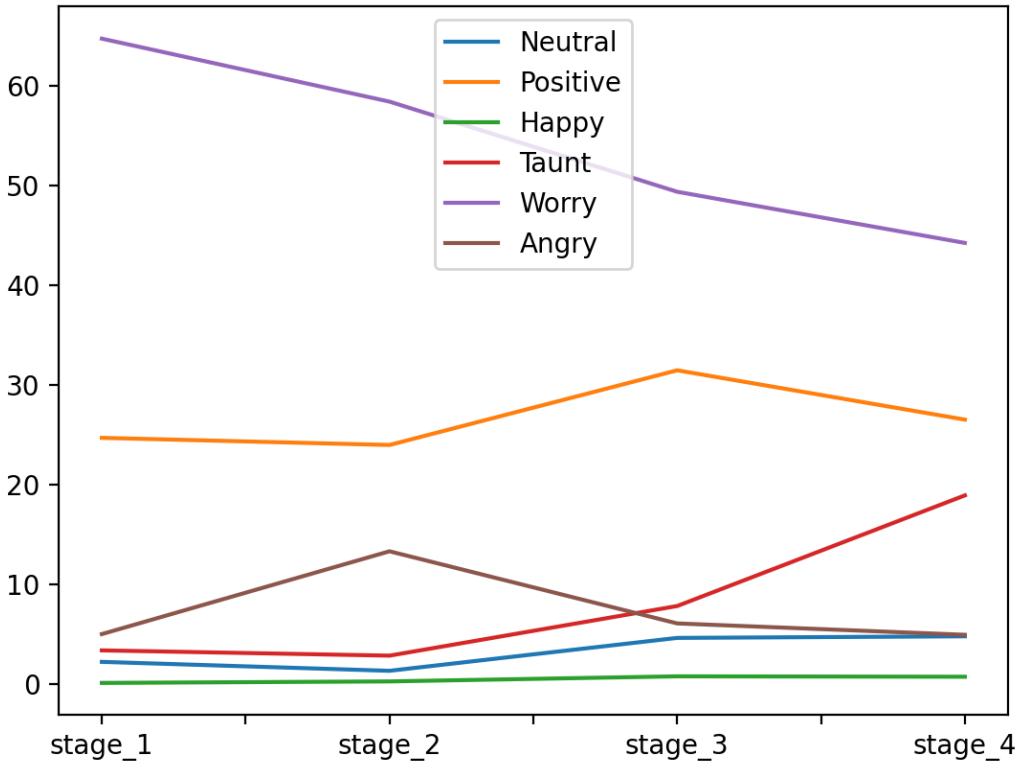


4. 第四阶段：

- 中性评论： 1553 比例：4.8%
- 乐观评论： 8625 比例： 26.5%
- 喜悦评论：233 比例：0.7%
- 嘲讽评论：6152 比例：18.9%
- 忧虑评论：14391 比例：44.2%
- 愤怒评论：1602 比例： 4.9%



3.2 疫情期间情绪比例变化



3.3 分析总结

根据各个阶段的饼状图来看，每个阶段公众所突出表现的**主要情绪**都是**忧虑**，这很自然，在不确定性的突发公共安全事件中，公众在社交媒体更倾向于吐槽、发泄负面情绪。

根据所有阶段的折线图来看，显然随着时间推移，**公众的忧虑情绪在逐渐下降，乐观的情绪稳中有升，而耐人寻味的是嘲讽的情绪在第三阶段、第四阶段增速明显。**可见一方面**我国政府的抗疫成果显著**，公众们的负面情绪在下降，正面情绪在上升，同时与境外的抗疫形成鲜明对比，后期境外疫情的失控，也让公众在社交媒体上感受了一场隔岸观火的乐趣，嘲讽的情绪的比例愈发升高。