# 人民网留言板-数据分析

## 数据爬取

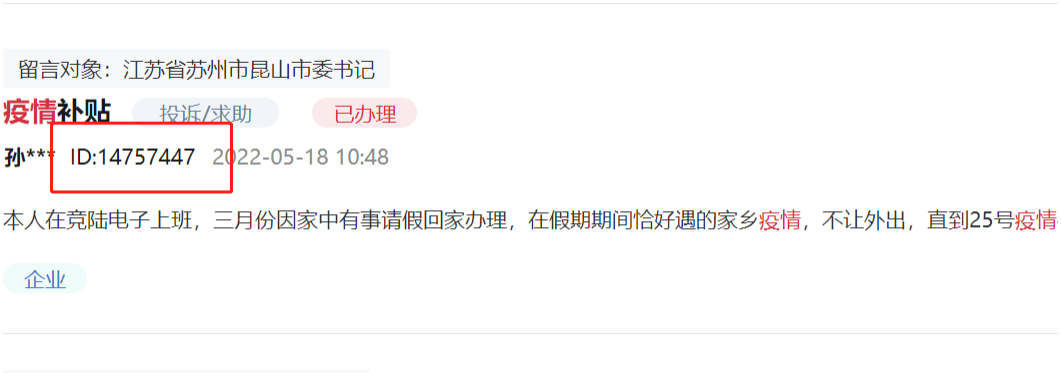
首先人民网是有反爬虫机制的，里面设置了对应的密钥，所以采用常规爬虫是无法爬取的，这时候我们就要采取另类措施了

这里我们采用的是selenium技术去实现自动化爬取内容

所谓自动化就是模拟人的行为去对网页进行数据采集

这里是对应的官方文档介绍：https://selenium-python-zh.readthedocs.io/en/latest/getting-started.html


这里的操作一共分为两步走，首先先去留言板获取对应的ID




然后再根据这些ID去伪造对应的URL


然后再去通过伪造的这个URL去获取对于的标签和内容

最后根据5个关键词一共获取1147条正文内容

其中领导有回复到的帖子一共为1070条内容

# 数据获取到的下一步就是数据处理工作

这里首先是把获取到两个文件进行合并，这里采用的是pandas中的concat函数去对文件进行合并，然后去进行去重drop_duplicates，再把多余的空值全部给删除，最后保留一个完整的文件


这个便是完整的文件

# 接着进行数据分析工作，先进行词云以及NLP分析

首先我们进行的是词云处理，这里一共分两步走，一是去分析提问里面的词云，二是分析官方回复里面的词云

这里我们首先需要一个停用词表，停用词表的意义就是帮助我们筛除一些无意义的词，也就是无效词和标点符号


接着我们根据jieba进行分词处理，分好词之后，我们写一个中文函数，也就是判断这个词是否为中文，如果是中文就保留下来，不是的话就去除，这个便是中文判断函数

```
def is_all_chinese(strs):
    for _char in strs:
        if not '\u4e00' <= _char <= '\u9fa5':
            return False
    return True
```

把词处理好之后，这里我们采用的stylecloud这个词云库，帮助我们有效的将词处理成词云图


接着我们再去把TOP200的词用pandas保存下来，放在data文件夹里面


然后我们再去对文本进行情感分析，这里情感分析我们采用的是百度开源成熟的NLP模型库，paddlehub

对应的文档介绍：https://www.paddlepaddle.org.cn/hubdetail?name=senta_bilstm&en_category=SentimentAnalysis

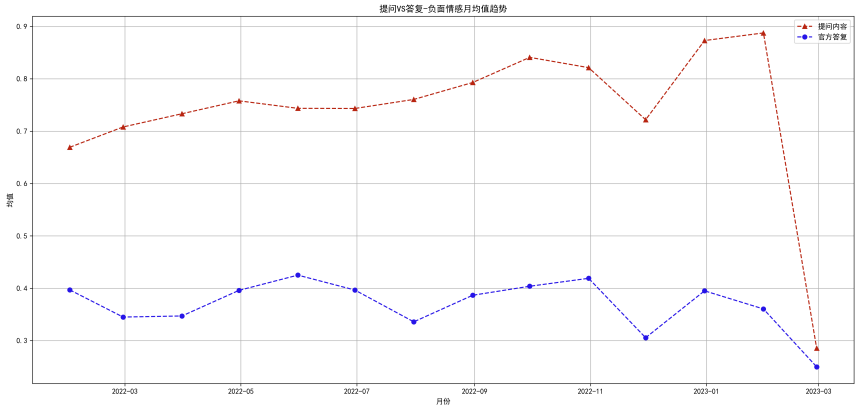
通过这个库去对文本内容进行打分

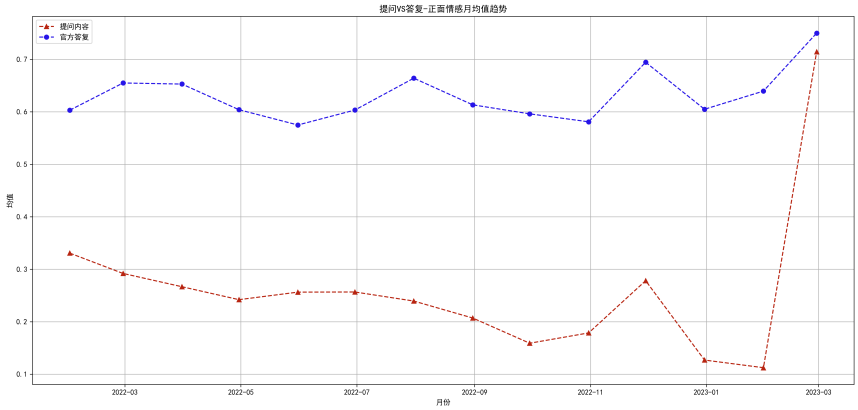
接着我们再去把我们的得到的情感得分用可视化的方式展示出来

首先我们把时间转化为序列，这里采用了pandas中的parse_dates方式，把时间作为序列，接着采用resample的方法把时间调为月份

接着我们采用matplotlib的方式进行可视化处理，最后生成的图片为：

从这里我们可以很好的看出，每个月情感的趋势是怎么样的一个情况




# 接着对文本进行一个聚类处理

聚类参考文献：https://zhuanlan.zhihu.com/p/78798251

首先我们还是先分词，把无效词给除去掉，

然后我们先把文本中的词转化为矩阵，这里采用的是sklearn中的CountVectorizer

接着再去统计每个词中的tf-idf权重，这里采用的是sklearn中的TfidfTransformer

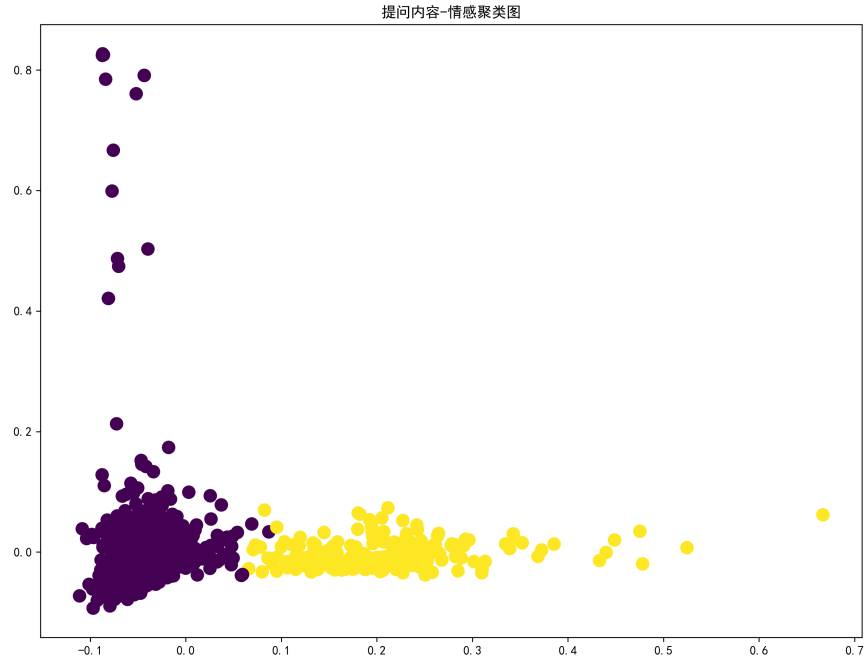
然后根据计算出来的权重和矩阵，进行平滑处理，也就是sklearn中的fit_transform

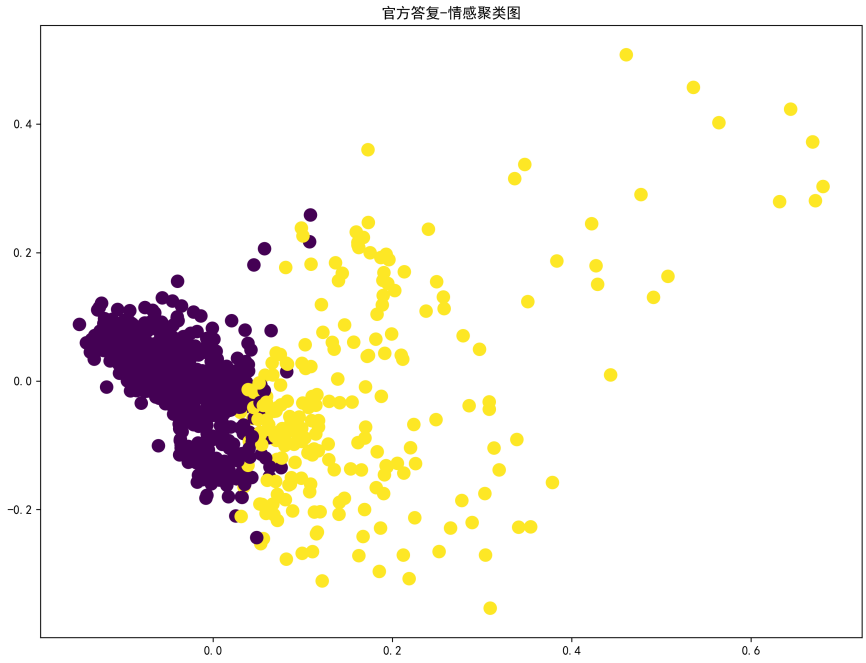
最后得出来的数据，我们再采用sklearn中的KMeans进行聚类

聚类结束后，我们先用PCA模块进行降维处理

接着再用matplotlib可视化出来，得出两个聚类后的结果展示图，这两个聚类，可以简单看着一个是非负，一个是负面，它们聚类的效果是怎么样的，这里只能清晰的判断，聚类的效果如何

并不能作为其他评判方式




# 接着我们再进行LDA主题建模

LDA参考文献： https://zhuanlan.zhihu.com/p/75222819

https://zhuanlan.zhihu.com/p/76636216

这里我们的步骤和上面和一样

首先我们还是先分词，把无效词给除去掉

接着我们开始构造主题数，寻找最优主题数，这里采用困惑度严格来说，判断标准并不合适，基于此我们这里采用的是另一种方式，也就是通过各个主题间的余弦相似度来衡量主题间的相似程度

> **(2) 寻找最优主题数**
>
> 基于相似度的自适应最优LDA模型选择方法，确定主题数并进行主题分析。实验证明该方法可以在不需要人工调试主题数目的情况下，用相对少的迭代，找到最优的主题结构。具体步骤如下。
>
> ① 取初始主题数k值，得到初始模型，计算各主题之间的相似度（平均余弦距离）。
>
> ② 增加或减少k值，重新训练模型，再次计算各主题之间的相似度。
>
> ③ 重复步骤②直到得到最优k值。
>
> 利用各主题间的余弦相似度来度量主题间的相似程度。从词频入手，计算它们的相似度，用词越相似，则内容越相近。
> 假定A和B是两个n维向量，A是，B是，则A与B的夹角θ的余弦值通过式（4）计算。
>
> $$P（w_j \mid d_j）=\sum_{s=1}^{K} P（w_i|z=s）\times P(z=s|d_j)（3）$$
>
> $$\cos\theta=\frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n}(A_i)^2 \sum_{i=1}^{n}(B_i)^2}=\frac{AB}{|AB|}（4）$$

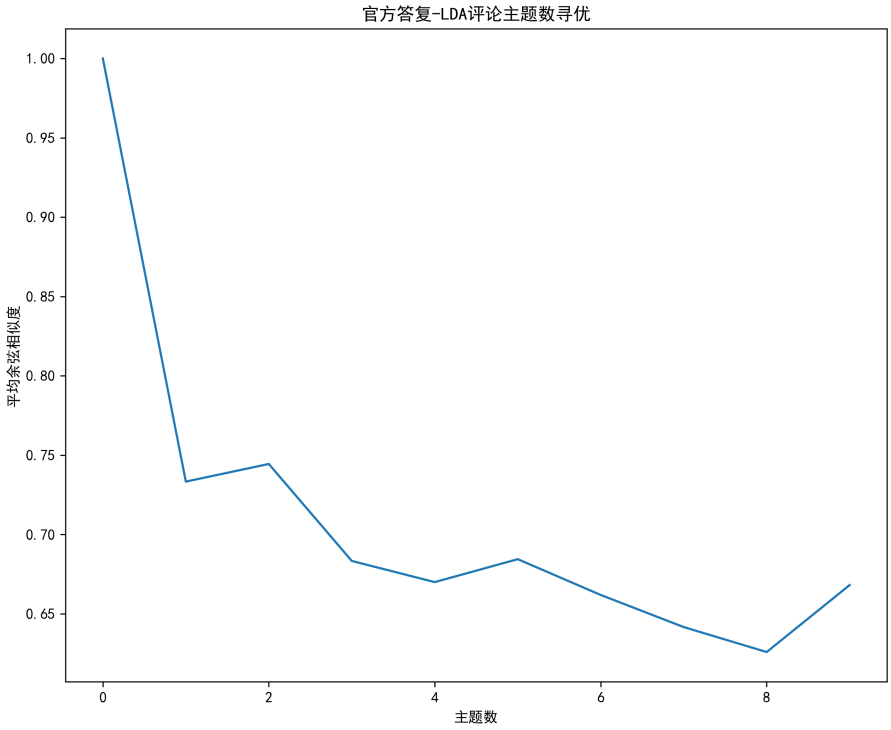
使用LDA主题模型，找出不同主题数下的主题词；每个模型各取出若干个主题词（比如前100个），合并成一个集合；生成任何两个主题间的词频向量；计算两个向量的余弦相似度，值越大就表示越相似；计算个主题数的平均余弦相似度，寻找最优主题数，如以下代码清单所示。

具体代码实现方式

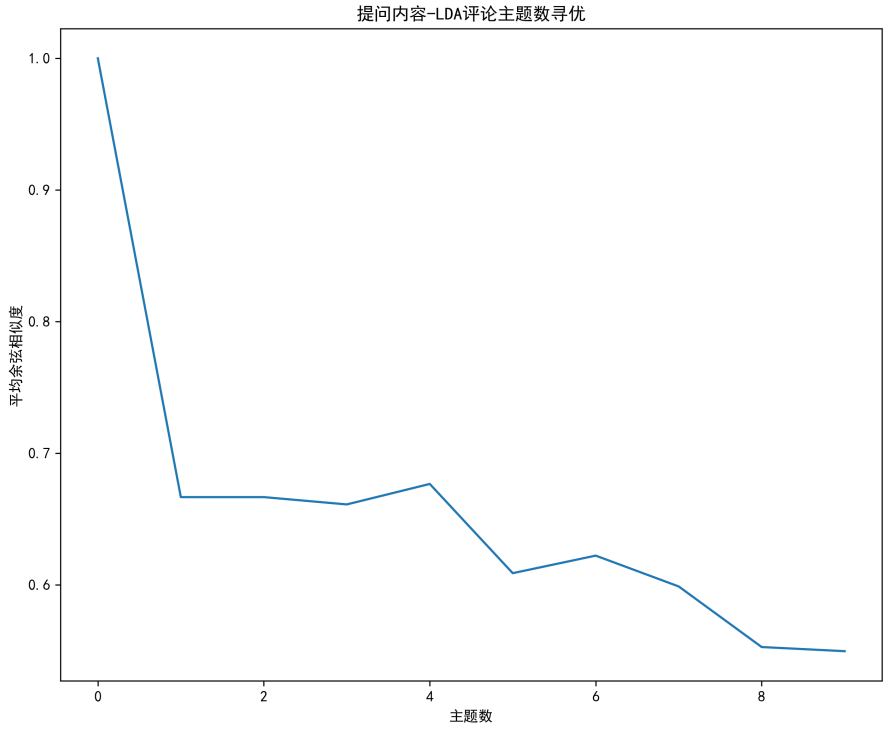
```
# 构造主题数寻优函数
    def cos(vector1, vector2):  # 余弦相似度函数
        dot_product = 0.0
        normA = 0.0
        normB = 0.0
        for a, b in zip(vector1, vector2):
            dot_product += a * b
            normA += a ** 2
            normB += b ** 2
        if normA == 0.0 or normB == 0.0:
            return (None)
        else:
            return (dot_product / ((normA * normB) ** 0.5))

        # 主题数寻优

    def lda_k(x_corpus, x_dict):
        # 初始化平均余弦相似度
        mean_similarity = []
        mean_similarity.append(1)

        # 循环生成主题并计算主题间相似度
        for i in np.arange(2, 11):
```

```
            lda = models.LdaModel(x_corpus, num_topics=i,
id2word=x_dict)  # LDA模型训练
            for j in np.arange(i):
                term = lda.show_topics(num_words=30)

            # 提取各主题词
            top_word = []
            for k in np.arange(i):

                top_word.append(['',join(re.findall('"(.*)"', i)) \
                                 for i in term[k][1].split('+')])  #
列出所有词

            # 构造词频向量
            word = sum(top_word, [])  # 列出所有的词
            unique_word = set(word)  # 去除重复的词

            # 构造主题词列表，行表示主题号，列表示各主题词
            mat = []
            for j in np.arange(i):
                top_w = top_word[j]
                mat.append(tuple([top_w.count(k) for k in
unique_word]))

            p = list(itertools.permutations(list(np.arange(i)), 2))
            l = len(p)
            top_similarity = [0]
            for w in np.arange(l):
                vector1 = mat[p[w][0]]
                vector2 = mat[p[w][1]]
                top_similarity.append(cos(vector1, vector2))

            # 计算平均余弦相似度
            mean_similarity.append(sum(top_similarity) / l)
        return (mean_similarity)
```

处理好之后，再通过matplotlib来进行作图

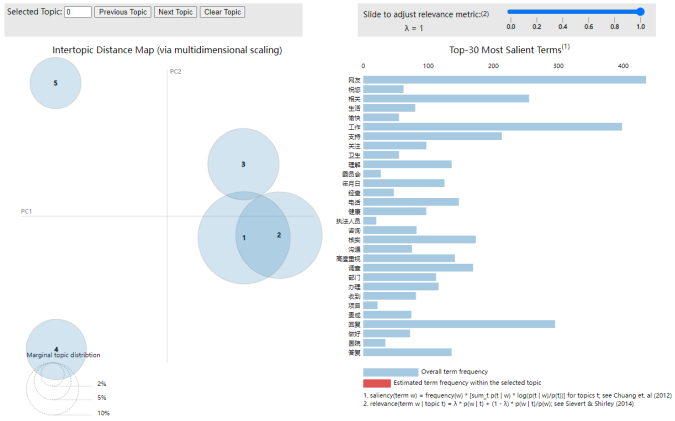


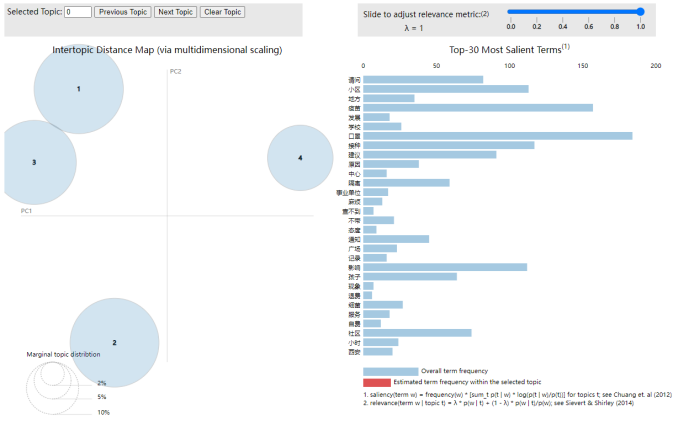

通过这种清晰的拐点，我们可以很好的判断对应的主题数是多少，例如官方答复这里，5是一个很好的拐点，所以我们可以采用5作为我们的官方答复的主题数




这里提问内容的拐点则是4

处理好之后，我们再采用gensim中的LdaModel去构建主题模型，最后把该主题用可视化的方式展示就是如图所示




接着我们再用gensim中的get_document_topics模块对我们的主题进行判断

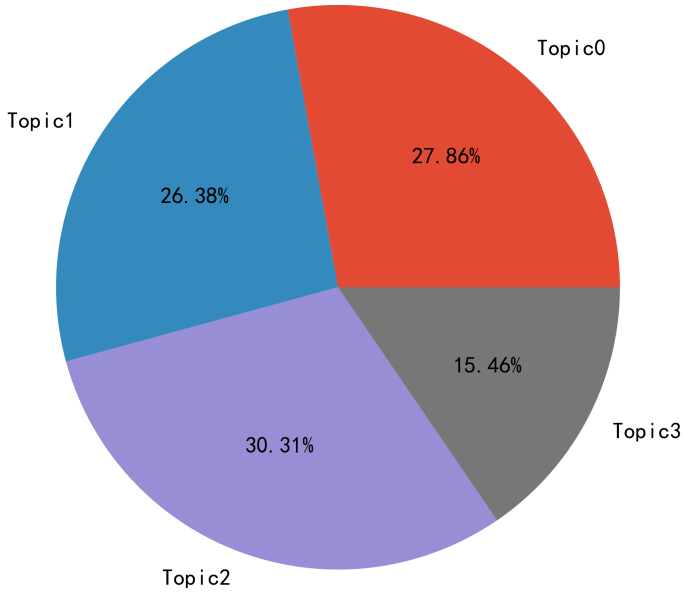
对每个对应的文本的主题数计算出来

C2 | 1

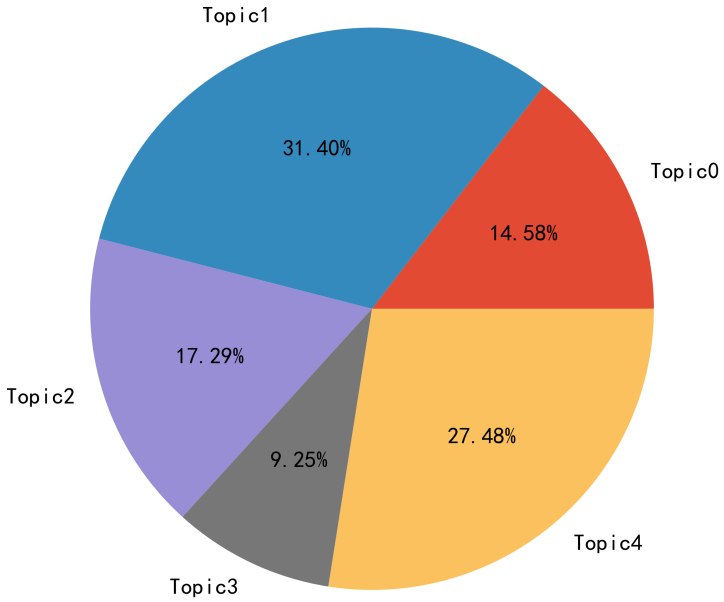
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 内容 | 主题概率 | 主题类型 | | | | | | |
| 2 | 希望加大横 | [(0, 0.0153 | 1 | | | | | | |
| | 自先感谢疫情期间各级同志们辛苦的付出，你们的责任感和关心百姓看得到。随着疫情的发展，病毒的病理本身也在发生变化。但除了防疫政策和现状数据的发布，对病毒本身科 | [(1, 0.9749 | 1 | | | | | | |

在得到它们的每个文档的主题数后，我们再用matplotlib进行饼图可视化出来

## 提问内容-主题强度




## 官方答复-主题强度




| 所属主题 | 文章数量 | 特征词 | 主题强度 |
|---|---|---|---|
| Topic0 | 156 | 工作、网友、相关、支持、收悉 | 0.14579 |
| Topic1 | 336 | 回复、网友、工作、情况、收悉 | 0.31402 |
| Topic2 | 185 | 网友、工作、情况、相关、高度 | 0.1729 |
| Topic3 | 99 | 网友、工作、相关、情况、回复 | 0.092523 |
| Topic4 | 294 | 网友、工作、防控、情况、收悉 | 0.27477 |

同时也把每个主题，它们的特征词给找出来

| A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|
| Topic0-主题 | Topic0-权重 | Topic1-主题 | Topic1-权重 | Topic2-主题 | Topic2-权重 | Topic3-主题 | Topic3-权重 | | | |
| 防控 | 0.008 | 口罩 | 0.011 | 疫苗 | 0.01 | 小区 | 0.006 | | | |
| 影响 | 0.007 | 建议 | 0.006 | 口罩 | 0.007 | 疫苗 | 0.005 | | | |
| 希望 | 0.006 | 小区 | 0.005 | 防控 | 0.007 | 病毒 | 0.005 | | | |
| 病毒 | 0.006 | 希望 | 0.005 | 接种 | 0.007 | 口罩 | 0.004 | | | |
| 口罩 | 0.005 | 人员 | 0.005 | 请问 | 0.006 | 接种 | 0.004 | | | |
| 核酸 | 0.005 | 病毒 | 0.005 | 希望 | 0.006 | 希望 | 0.004 | | | |
| 疫苗 | 0.005 | 防控 | 0.005 | 人员 | 0.005 | 隔离 | 0.004 | | | |
| 居民 | 0.004 | 疫苗 | 0.004 | 工作 | 0.005 | 工作 | 0.004 | | | |
| 人员 | 0.004 | 社区 | 0.004 | 健康 | 0.004 | 防控 | 0.003 | | | |
| 接种 | 0.004 | 工作 | 0.004 | 谢谢 | 0.004 | 谢谢 | 0.003 | | | |
| 检测 | 0.004 | 防疫 | 0.004 | 小区 | 0.004 | 时间 | 0.003 | | | |
| 孩子 | 0.004 | 核酸 | 0.003 | 病毒 | 0.004 | 社区 | 0.003 | | | |
| 政策 | 0.004 | 部门 | 0.003 | 核酸 | 0.004 | 相关 | 0.003 | | | |
| 部门 | 0.004 | 风险 | 0.003 | 影响 | 0.004 | 通知 | 0.003 | | | |
| 建议 | 0.004 | 影响 | 0.003 | 检测 | 0.004 | 影响 | 0.003 | | | |
| 小区 | 0.003 | 政府 | 0.003 | 政策 | 0.003 | 人员 | 0.003 | | | |
| 防疫 | 0.003 | 政策 | 0.003 | 医院 | 0.003 | 孩子 | 0.003 | | | |
| 工作 | 0.003 | 情况 | 0.003 | 地方 | 0.003 | 原因 | 0.003 | | | |
| 生活 | 0.003 | 发现 | 0.003 | 解决 | 0.003 | 年月日 | 0.002 | | | |
| 书记 | 0.003 | 你好 | 0.003 | 防疫 | 0.003 | 请问 | 0.002 | | | |

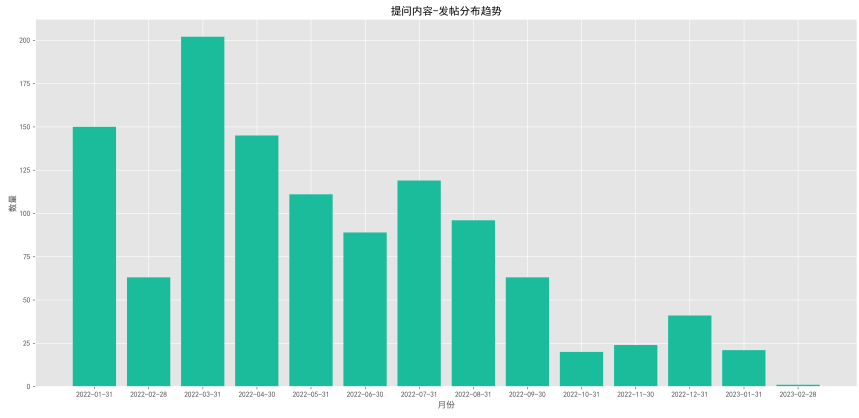
以及不同主题，它们top20的词的权重给对应计算出来

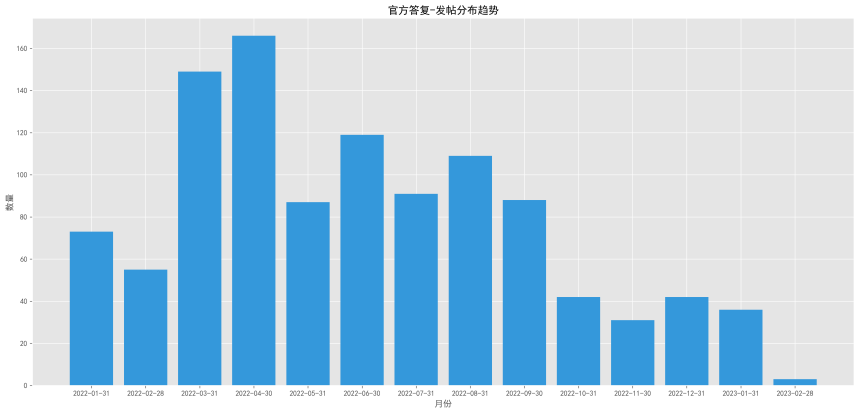
# 最后就是根据上面获取到的内容，进行一些可视化

首先我们这里根据获取的文本内容，统计一下，每个月的推文是多少

这里就是根据文档，然后先是用pandas中的parse_dates去把时间转为序列，接着对序列进行月份统计

最后再用resample得出每个月份的推文数量是多少，接着根据这些的数据用matplotlib来进行可视化处理





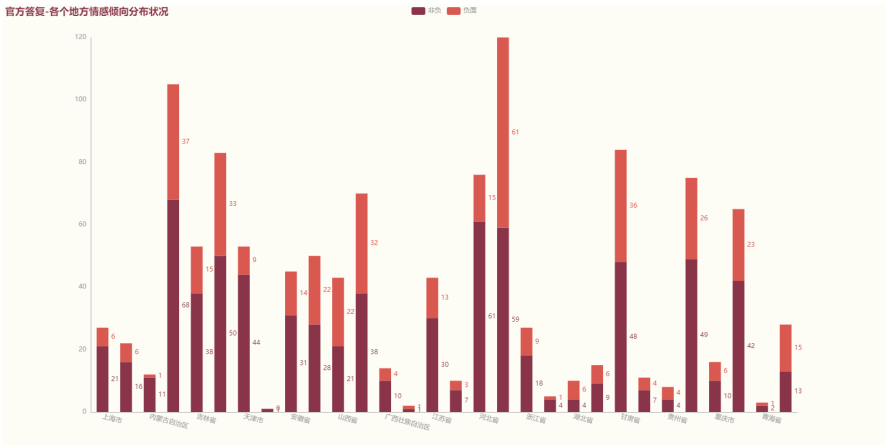

接着我们需要得知，每个地方他们的情感占比是一个什么样的情况

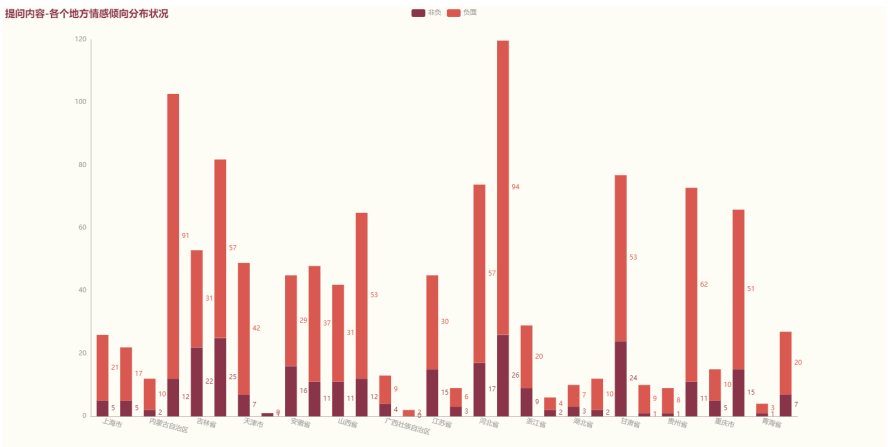
这里我们先是把当地转化为对应的省份、市、自治区

这里就是自己编写对应的清洗函数，接着调用pandas中的apply模块，把数据清洗好之后

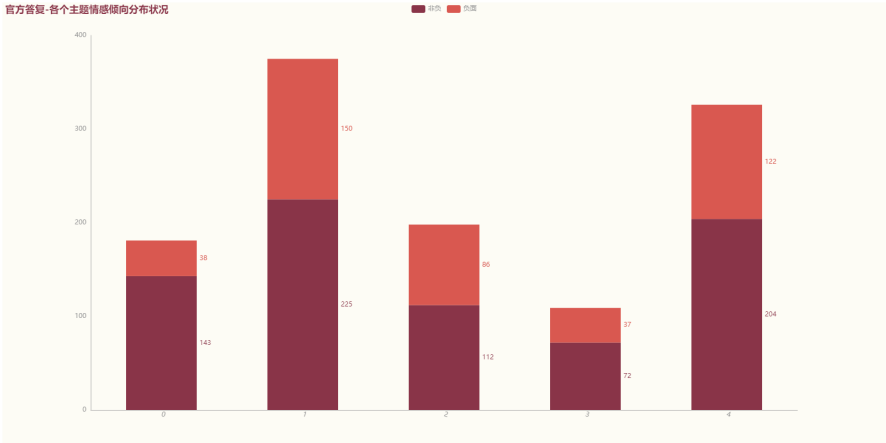
我们再用pandas中的groupby模块把地区进行归类统计

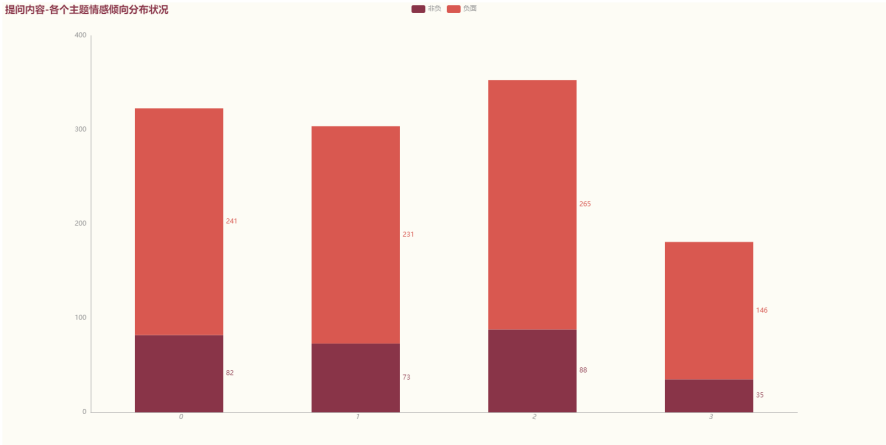
最后得到的数据，我们再用pyecharts进行可视化出来



而后，主题的情感统计和上面的思路一样，采用的技术也是一样，只是个别地方需要修改代码




这里有一点是需要注意的，因为中立其实有点难以判断，所以一般业内统计都是归为非负和负面这两类

# 以上便是全部分析的思路和方法