该项目一共分四步走

1.首先进行数据处理

数据处理，我们用到的是pandas常用库

先进行去重drop_duplicates，把重复的内容处理好之后，我们开始删除一些无效的内容，例如表情包，无效词等

这里首先就是先去掉表情包，然后再判断该文本是否为中文，接着再去用停用词文本，去除无效词

```
def emjio_tihuan(x):
    x1 = str(x)
    x2 = re.sub('(\[.*?\])', "", x1)
    x3 = re.sub(r'@[\w\u2E80-\u9FFF]+:?/\[\w+\]', '', x2)
    x4 = re.sub(r'\n', '', x3)
    return x4

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

def get_cut_words(content_series):
    # 读入停用词表
    stop_words = ["请问",'真的',"支持","不用","昨天","只能","感觉","谢谢","安排","你好","告诉","好像","带你去","这是","一点","明天","刚刚","一年","东西","不到","我刚","不上"]

    with open("stopwords_cn.txt", 'r', encoding='utf-8') as f:
        lines = f.readlines()
        for line in lines:
            stop_words.append(line.strip())
```

接着我们stylecloud进行词云图，这样方便我们看看整体的分词效果如何，是否有一些词要不要去掉

最后的结果如下：

把结果都处理好了就开始NLP，情感分析

2、情感分析

文本挖掘中，情感分析是经常需要使用到，而进行主题模型分析之前，对数据集进行文本分类再进行分析具有必要性，因为分类以后，每一类的主题才会更明显。而snownlp是一个python写的类库，可以方便的处理中文文本内容

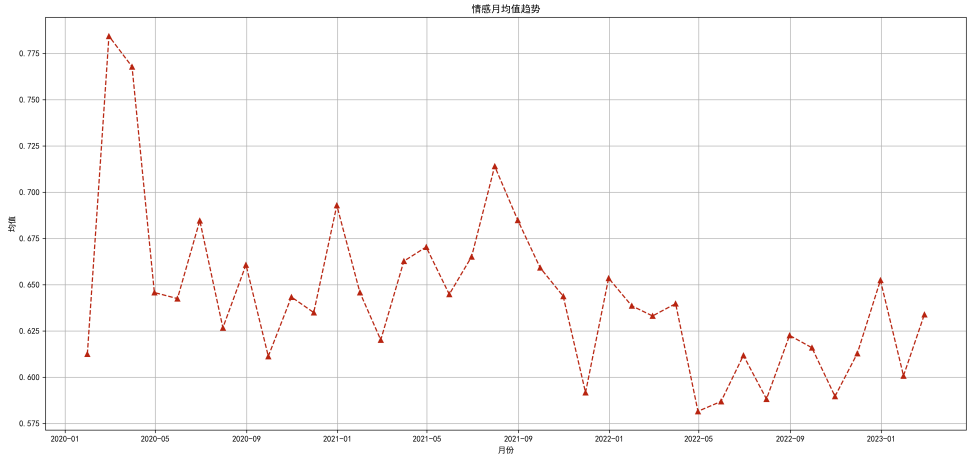
这里我们还是根据上面的方法，先进行分词处理，因为这样更有效帮助机器进行文本判断，从而给出正确的评价，这里有一点是要注意的

就是snownlp毕竟是广义的，无法做到精准判断该文本的正确评分，只能给出大概的数值，这也是所有机器学习的通病，毕竟是机器，NLP还是主观占比居多，所以提供的数值仅供参考

	序号	来源	抖音昵称	抖音号	UID	sec_uid	IP归属	认证	点赞	回复	评论	评论时间	情感分数
2	0	1	https://ww	看见熊美	3.42E+09	6.77E+14	MS4wLjABA		厦门市集美	42735	2226	在价格上，2020-04-2	0.75207
3	1	2	https://ww	用户53280	2.79E+09	3.81E+15	MS4wLjABA			7544	763	安徽有没有2020-04-2	0.775364
4	2	3	https://ww	囧妞儿呀	1.11E+08	7.39E+10	MS4wLjABA			5722	583	河南有吗 2020-04-2	0.296296
5	3	4	https://ww	伍.月.	3.81E+08	8.22E+10	MS4wLjABA			4058	221	河南南阳丰2020-04-2	0.033313
6	4	5	https://ww	繁华落尽	1.24E+09	1E+11	MS4wLjABA			2983	82	云南哪里可2020-04-2	0.545455
7	6	7	https://ww	人生如梦	2.19E+09	6.2E+10	MS4wLjABA			1108	41	湖南有吗 2020-04-2	0.555556
8	7	8	https://ww	用户49677	5.82E+08	2.22E+10	MS4wLjABA			352	24	河南大郑州2020-04-2	0.490384
9	10	11	https://ww	槐夏逐浪	1.1E+09	1.01E+11	MS4wLjABA			60	17	福建年底有2020-04-2	0.515758
10	11	12	https://ww	骨科健将洋	91318818	6.82E+10	MS4wLjABA			77	0	真不错 2020-04-2	0.863252
11	12	13	https://ww	济南楼市李	2.84E+09	4.09E+15	MS4wLjABA		济南蒲康所	90	4	我就想知道2020-04-2	0.908795
12	15	16	https://ww	素颜照	6.17E+08	5.97E+10	MS4wLjABA			1	1	请问厦门后2021-09-0	0.912378
13	16	17	https://ww	怎么又胖了	3.44E+09	3.61E+15	MS4wLjABA			1	0	怎么预约 2021-11-3	0.333333
14	18	19	https://ww	Z　z	5.89E+08	9.52E+10	MS4wLjABA			0	0	你好，怎么2022-03-0	0.333333
15	20	21	https://ww	阿妹	79625663	7.01E+10	MS4wLjABA			0	0	怎么预约 2021-12-0	0.333333
16	21	22	https://ww	心静安然	1.9E+09	1.09E+11	MS4wLjABA			0	0	这个怎么打2021-11-2	0.333333
17	22	23	https://ww	114374821	1.14E+08	7.43E+10	MS4wLjABA			0	0	厦门怎么五2021-11-2	0.543982
18	23	24	https://ww	余生安好	3.44E+09	9.23E+14	MS4wLjABA			0	0	厦门要在哪2021-11-2	0.543982
19	26	27	https://ww	仙女	76811150	6.96E+10	MS4wLjABA			0	0	已打，三针2021-10-0	0.890175
20	27	28	https://ww	若不见	6.25E+08	8.59E+10	MS4wLjABA			0	0	在哪里预约2021-09-0	0.333333
21	28	29	https://ww	小李好物分	5.97E+08	9.55E+10	MS4wLjABA			0	0	打开抖音发2021-08-2	0.370378
22	29	30	https://ww	小猪暴富吧	1.07E+08	7.35E+10	MS4wLjABA			0	0	谷谈医院手2021-08-1	0.417629
23	30	31	https://ww	鹿97啊	1.78E+09	1.08E+11	MS4wLjABA			1	0	猪爷时主2021-07-3	0.588866
24	31	32	https://ww	臭豆腐	1.19E+09	1.02E+11	MS4wLjABA			0	0	广西有吗 2021-07-2	0.75
25	32	33	https://ww	52h	1.19E+09	7.21E+10	MS4wLjABA			0	0	几价 2021-07-2	0.77168
26	33	34	https://ww	柚子茶	9.73E+08	1.4E+10	MS4wLjABA			0	0	怎么预约啊2021-07-0	0.333333
27	34	35	https://ww	青砖伴瓦泥	1.23E+09	1.02E+11	MS4wLjABA			1	0	有九价的吗2021-07-0	0.840619
28	35	36	https://ww	小鲜椒	8.55E+08	6.15E+10	MS4wLjABA			1	0	这个怎么打2021-06-1	0.333333
29	36	37	https://ww	Y	2.8E+09	2.21E+15	MS4wLjABA			0	0	有年龄限表2021-06-1	0.707692
30	37	38	https://ww	佟佟	2.7E+08	6.19E+10	MS4wLjABA			0	0	去哪个医院2021-06-0	0.421017
31	38	39	https://ww	梦蝉	2.7E+08	8.81E+10	MS4wLjABA			2	0	深圳怎么当2021-05-1	0.380952
32	39	40	https://ww	Z	1.58E+09	1.46E+10	MS4wLjABA			0	1	厦门那个医2021-04-0	0.793983
33	40	41	https://ww	旧人丶	3.82E+09	1.83E+15	MS4wLjABA			0	0	广西南宁医2021-03-2	0.635404
34	41	42	https://ww	叶子、	3.81E+08	9.39E+10	MS4wLjABA			0	0	超打这个医2021-03-2	0.656384
35	42	43	https://ww	平静的心	1.06E+09	8.54E+10	MS4wLjABA			0	0	怎么预约啊2021-03-2	0.676398
36	43	44	https://ww		3.88E+09	1E+11	MS4wLjABA			0	0	位置在哪里2021-03-2	0.656384
37	44	45	https://ww	嗯啊嗯	8.75E+08	9.77E+10	MS4wLjABA			0	0	这个位置在2021-03-2	0.656384
38	45	46	https://ww	浩浩	1.99E+08	7.69E+10	MS4wLjABA			0	0	不错不错，2021-03-2	0.994863
39	47	48	https://ww	木丙	3.75E+09	1.93E+15	MS4wLjABA			0	0	那这个就对2021-03-2	0.822518
40	48	49	https://ww	奶里	2.37E+09	8.92E+14	MS4wLjABA			0	0	这是拯救了2021-03-2	0.917102

Sheet1

接着我们获取到对应的分值之后，我们可以根据数据来做一个时间趋势图，从而得知，在每个月的一个分值走向，这里才去的是均值处理，把每个月的所有分值相加求平均值，所以这里还是有一定的参考价值，可以作为正确的评判标准，从这里我们可以得知，在2020-01： 2020-04,情绪都较为积极比较偏正向，这里分值是从0-1直接的，接近0则是负面，接近1则是正面



3、TF-IDF计算

1.TF-IDF计算

TF-IDF（Term Frequency-InversDocument Frequency）是一种常用于信息处理和数据挖掘的加权技术。该技术采用一种统计方法，根据字词的在文本中出现的次数和在整个语料中出现的文档频率来计算一个字词在整个语料中的重要程度。它的优点是能过滤掉一些常见的却无关紧要本的词语，同时保留影响整个文本的重要字词。计算方法如下面公式所示：

$$\mathrm{TF}-\mathrm{IDF}=\mathrm{TF}*\mathrm{IDF}$$

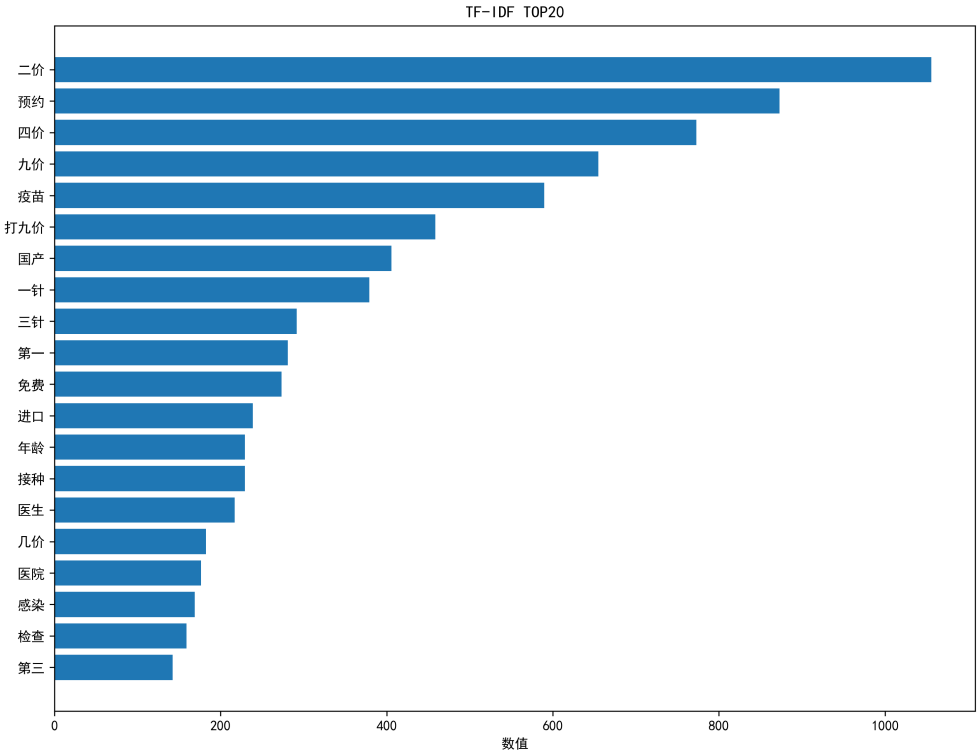
TF（Term Frequency）表示某个关键词在整篇文章中出现的频率。IDF（InversDocument Frequency）表示计算倒文本频率。文本频率是指某个关键词在整个语料所有文章中出现的次数。倒文档频率又称为逆文档频率，它是文档频率的倒数，主要用于降低所有文档中一些常见却对文档影响不大的词语的作用。

主要实现代码在这一块

```
    # 将文本中的词语转换为词频矩阵 矩阵元素a[i][j] 表示j词在i类文本下的词频
    vectorizer = CountVectorizer()

    # 该类会统计每个词语的tf-idf权值
    transformer = TfidfTransformer()

    # 第一个fit_transform是计算tf-idf 第二个fit_transform是将文本转为词频矩阵
    tfidf = transformer.fit_transform(vectorizer.fit_transform(corpus))
    # 获取词袋模型中的所有词语
    word = vectorizer.get_feature_names()

    # 将tf-idf矩阵抽取出来 元素w[i][j]表示j词在i类文本中的tf-idf权重
    weight = tfidf.toarray()

    data = {'word': word,
            'tfidf': weight.sum(axis=0).tolist()}
    df2 = pd.DataFrame(data)
```

```
    df2['tfidf'] = df2['tfidf'].astype('float64')
    df2 = df2.sort_values(by=['tfidf'], ascending=False)
```

我们从该文档中，抽取前20的词，会发现这里的词和高频词是一样的，但是它们的权重却又是不一样的，这就是高频词和TF-IDF之间的差异了



4、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。

$$P（w_j \mid d_j）= \sum_{s=1}^{K} P（w_i|z=s）\times P(z=s|d_j)（3）$$

$$\cos\theta = \frac{\sum_{i=1}^{n} A_i B_i}{\sum_{i=1}^{n}(A_i)^2 \sum_{i=1}^{n}(B_i)^2} = \frac{AB}{|AB|}（4）$$

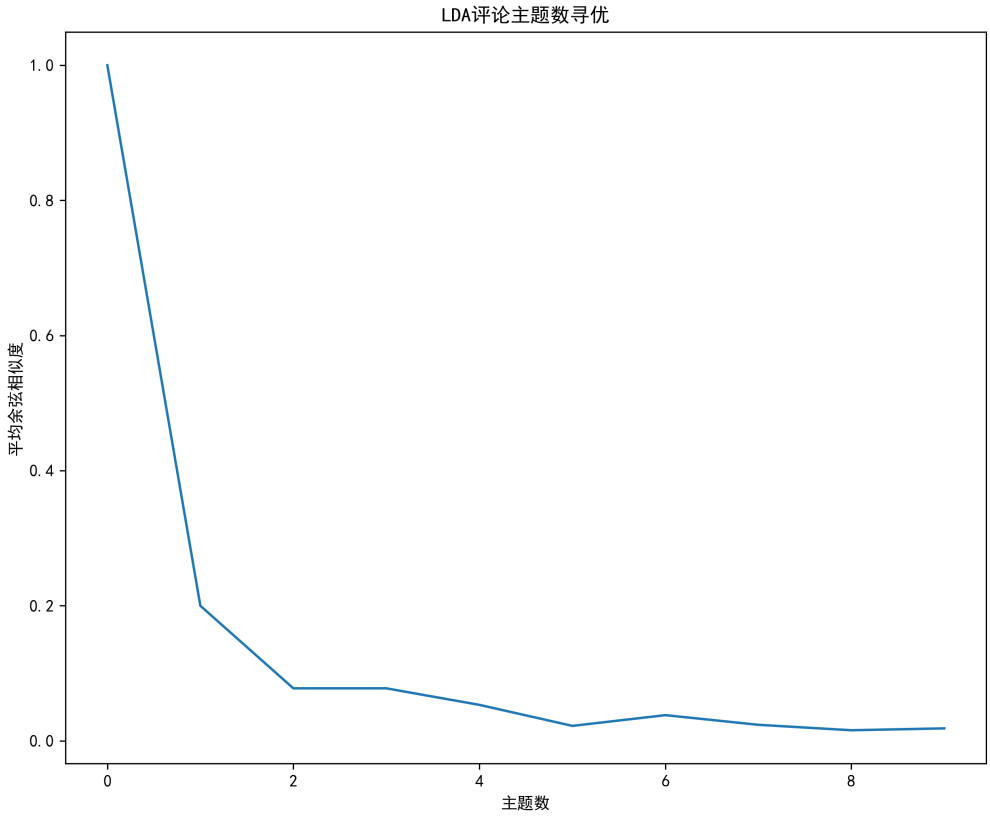
使用LDA主题模型，找出不同主题数下的主题词；每个模型各取出若干个主题词（比如前100个），合并成一个集合；生成任何两个主题间的词频向量；计算两个向量的余弦相似度，值越大就表示越相似；计算个主题数的平均余弦相似度，寻找最优主题数，如以下代码清单所示。

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，在这里明显在5的时候处于谷底，因此根据这一点，我们可以选取主题数为5



从而进行建模，最后呈现的效果图如下：

可以通过点击不同的圆圈，来查看不同主题下，不同主题词的权重