基于主题的文本情感分析实验报告

组内成员	学号	负责任务
梁济凡	17341091	写代码
林正青	17341103	写代码
林楠	17341101	写报告
刘异橦	17341112	做PPT

实验题目

关于基于主题的文本情感分析。

实验内容

针对DataFountain的一个比赛来进行关于主题的情感分析，由于比赛已经结束，所以本实验仅基于比赛数据来进行一定的尝试和拓展。根据比赛提供的将近20000条数据来进行相关的情感分析。

实验总体步骤是先对数据进行读入->分词处理->计算词频->关键词计算（TF-IDF分析词的重要性）->计算文本相似度->K-means聚类->谱系聚类->nltk分词处理->过滤停用词->利用scipy做谱系聚类->主题建模->预测文档对应的主题->主题对应的关键词->模型可视化->情感分析

其中最为重要的实验步骤是数据读入->分词处理->计算词频->TF-IDF分析词的重要性->nltk分词处理->过滤停用词->主题建模->预测文档对应的主题->主题对应的关键词->模型可视化->情感分析

其余的一些都是拓展内容，从不同的方面尝试情感分析的一些内容。

实验结果

数据读入

利用python的pandas库来进行csv文件数据的读入

```
import pandas as pd
df = pd.read_csv('train_content.csv',encoding='utf-8')
df
df = df[['Data']]
df
```

	Data
0	收到了，太实惠了，买了一大箱，以后继续购买，送货速度快服务也好
1	热水器加热时间太长，安装费太贵，预留太阳能口摆设，根本用不到，没有水位指示器，加满热水的指示...
2	真**差，我都无语了。眼瞎才买了
3	很满意，因为出差，还没有用，东西装上我看了，非常好
4	给奶奶买的，效果不错，厂家服务也好，给好评[追评]
5	还没用，等用了再来追评吧
6	这个中奖有用吗？且不说中奖，这个是我买了这么多一加3t钢化膜中最高清最好看的，没有之一。
7	不错，家用刚刚好，制冷也不错
8	电流声真心受不了！而且电脑开机的时候会有爆破声。。。
9	电视挺好，都挺满意

分词处理

现在已有的分词算法大体可以分为

- 基于字符串匹配的分词方法：这种做法需要有一个足够大的可匹配词典，这个词典中存储了很多的词条，然后计算机会与这些已经存在的词条进行匹配，匹配到一个就完成一个分词。
- 基于理解的分词方法：其做法就是让计算机尽可能模拟人类对于句子的理解，在分词的时候让计算机做到句法分析，语义分析，但是因为汉语言的庞杂以及多变性，这种分词方法并没有那么成熟。
- 基于统计的分词：这种做法是给出大量的已分好词的文本，然后利用机器学习此文本的分词方式和方法，训练相应的分词模型，从而达到对未知文本的分词，随着大量的语料库的出现以及应用，这种基于统计的分词渐渐的崭露头角。

经过一些文本分析的资料，决定使用结巴分词的方法来对自然语言进行处理。

结巴分词

结合基于规则和基于统计

- 基于前缀词典实现高效的词图扫描，生成句子中汉字所有可能成词情况所构成的有向无环图 (DAG)
- 采用动态规划查找最大概率路径, 找出基于词频的最大切分组合

下图是相关的处理方法

In [8]:
```
titles = df['Data']#提取需分词的内容
#titles.head(10)

#按词性分词
import jieba.posseg as pseg
titles1=[]
for title in titles:
    title1=pseg.cut(title)
    title2=[[word.word,word.flag]for word in title1]
    titles1.append(title2)
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\74265\AppData\Local\Temp\jieba.cache
Loading model cost 1.349 seconds.
Prefix dict has been built succesfully.
```

In [9]:
```
#titles1
print(titles1[1])
```

```
[['热水器', 'n'], ['加热', 'v'], ['时间', 'n'], ['太', 'd'], ['长', 'a'], [',', 'x'], ['安装费', 'n'], ['太贵', 'nr'], [',', 'x'], ['预留', 'v
设', 'v'], [',', 'x'], ['根本', 'a'], ['用', 'p'], ['不到', 'v'], [',', 'x'], ['没有', 'v'], ['水位', 'n'], ['指示器', 'n'], [',', 'x'], ['加
['指示灯', 'n'], ['放在', 'v'], ['了', 'ul'], ['最', 'a'], ['侧面', 'f'], [',', 'x'], ['不', 'd'], ['方便', 'a'], ['用户', 'n'], ['看', 'v'], [
'd'], ['斜', 'v'], ['着', 'uz'], ['看', 'v'], ['才能', 'v'], ['看到', 'v'], [',', 'x']]
```

In [10]:
```
#按词性抽取
titles2=[[word[0] for word in title if word[1]=='n'
         or word[1]=='v' or word[1]=='a' or word[1]=='d' or word[1]=='nr'
         or word[1]=='nrfg' or word[1]=='ns' or word[1]=='nt' or word[1]=='nz'
         or word[1]=='vn' or word[1]=='vd' or word[1]=='an' or word[1]=='ad'
         or word[1]=='l']for title in titles1]
```

In [11]:
```
#按词性抽取
titles2=[[word[0] for word in title if word[1]=='n'] for title in titles1]
print(titles2[1])
```

```
['热水器', '时间', '安装费', '太阳能', '水位', '指示器', '热水', '指示灯', '用户', '指示灯']
```

In [12]:
```
#删除一个字的词
titles3 =[[word for word in text if len(word)>1]for text in titles2]
print(titles3[1])
```

```
['热水器', '时间', '安装费', '太阳能', '水位', '指示器', '热水', '指示灯', '用户', '指示灯']
```

In [13]:
```
titles3
```

Out[13]:
```
[['速度'],
 ['热水器', '时间', '安装费', '太阳能', '水位', '指示器', '热水', '指示灯', '用户', '指示灯'],
 ['眼睛'],
 [],
 ['奶奶', '效果', '厂家'],
 [],
 ['钢化'],
```

计算词频

最后计算词频结果为

	terms	count
50	感觉	1647
33	质量	1415
12	效果	1303
41	有点	1144
29	味道	883
38	问题	749
24	价格	690
103	手机	631
40	物流	629

将相关的词频数据导出到了allWordsCount.csv

由上图可看出是基于一些词性来进行分类并计算词频，详情可以参考代码。

关键词计算

根据网上的相关资料发现关键词计算一般使用

- TF-IDF
- textrank

两种方法,这里采用TF-IDF分析法

TF-IDF是一种用于信息检索与文本挖掘的常用加权技术。

在一份给定的文件里，**词频**（term frequency，tf）指的是某一个给定的词语在该文件中出现的频率。这个数字是对**词数**（term count）的归一化，以防止它偏向长的文件。（同一个词语在长文件里可能会比短文件有更高的词数，而不管该词语重要与否。）对于在某一特定文件里的词语t_i来说，它的重要性可表示为：

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

以上式子中$n_{i,j}$是该词在文件d_j中的出现次数，而分母则是在文件d_j中所有字词的出现次数之和。

逆向文件频率（inverse document frequency，idf）是一个词语普遍重要性的度量。某一特定词语的idf，可以由总文件数目除以包含该词语之文件的数目，再将得到的商取以10为底的对数得到：

$$\mathrm{idf_i} = \lg \frac{|D|}{|\{j : t_i \in d_j\}|}$$

其中

- |D|：语料库中的文件总数
- $|\{j : t_i \in d_j\}|$：包含词语t_i的文件数目（即$n_{i,j} \neq 0$的文件数目）如果词语不在数据中，就导致分母为零，因此一般情况下使用$1 + |\{j : t_i \in d_j\}|$

然后

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}$$

某一特定文件内的高词语频率，以及该词语在整个文件集合中的低文件频率，可以产生出高权重的tf-idf。因此，tf-idf倾向于过滤掉常见的词语，保留重要的词语。

> **文本相似度**
>
> **K-means聚类**
>
> **谱系聚类**

由于文本分析中**文本相似度**与 **K-means聚类**与 **谱系聚类**也经常需要使用所以这里进行了一定的尝试和拓展

k-means 聚类

聚类算法有很多种，K-Means 是聚类算法中的最常用的一种，算法最大的特点是简单，好理解，运算速度快，但是只能应用于连续型的数据，并且一定要在聚类前需要手工指定要分成几类。

K-Means 聚类算法的大致意思就是"物以类聚，人以群分"。

K-means 聚类

```
#利用scikit-learn中的Kmeans模型进行聚类
from sklearn.cluster import KMeans

num_clusters = 9

km = KMeans(n_clusters=num_clusters)

km.fit(tfidf_matrix)

clusters = km.labels_.tolist()
```

```
clusters[:10]#前10个文档的类
```

```
[2, 2, 2, 2, 0, 2, 2, 2, 2, 2]
```

```
df['cluster'] = clusters
```

```
E:\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-co
  if __name__ == '__main__':
```

```
df.tail()
```

	Data	t
19989	先说包装特别简陋，产品的 味道刺鼻，不管是洗面奶还沐浴露，洗完之后都有刺疼感，很疼的。你早吗...	产品
19990	一颗星都不想给，我跟客服反映是假货，客服说，不能拿商标的粗糙性来断定就是假货，也可能是各个地...	客服 假货 客服 商标 假货 地方 大家 商标
19991	提神，或者当爽肤水的用也挺好，关键还能避蚊虫	关键
19992	太小了，退了，不太适合摄影背	太小

根据相关分类可以看出k-means将类似的文本进行了聚类

K-Means处理后相关输出导出到了clusters.csv文件中

文本相似度和谱系聚类不是特别重要就简单看一些结果

文本相似度

```
#计算距离
from sklearn.metrics.pairwise import cosine_similarity
sim = cosine_similarity(tfidf_matrix)
dist = 1.0 - cosine_similarity(tfidf_matrix)#根据两个文档的词条的TF-IDF值来计算
```

```
sim[5,111]
```

```
0.0
```

```
[s for s in sim]
```

```
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0.        , 0.        , 0.        , ..., 0.        , 0.28400632,
       0.        ]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0.79026171, 0.        , 0.        , ..., 0.        , 0.        ,
       0.        ]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0., 0., 0., ..., 0., 0., 0.]),
array([0.        , 0.        , 0.        , ..., 0.        , 0.        ,
       0.60008395]),
array([0., 0., 0., ..., 0., 0., 0.]),
```

谱系聚类由于数据量过大，电脑内存不够了没有跑出来图。

主题建模结果

LDA主题模型

利用LDA主题模型,它可以将文档集中每篇文档的主题按照概率分布的形式给出。同时它是一种无监督学习算法，在训练时不需要手工标注的训练集，需要的仅仅是文档集以及指定主题的数量k即可。此外LDA的另一个优点则是，对于每一个主题均可找出一些词语来描述它。

58]:
```
#训练一个LDA模型
%time lda = models.LdaModel(corpus, num_topics=9, id2word=dictionary, update_every=5, alpha=0.1, eta=0.1
```

```
Wall time: 8.03 s
```

59]:
```
print(lda[corpus[0]]) #打印第一个文档的主题建模结果
```

```
[(0, 0.57892376), (1, 0.052633613), (2, 0.052632548), (3, 0.052632283), (4, 0.052633125), (5, 0.05264204
```

预测文档对应的主题

60]:
```
doctop = [ s for s in lda[corpus]]
doctop = pd.DataFrame(doctop)
doctop
```

60]:

	0	1	2	3	4	5	
0	(0, 0.57892394)	(1, 0.05263361)	(2, 0.052632548)	(3, 0.052632283)	(4, 0.052633125)	(5, 0.052641846)	(6, 0.05
1	(1, 0.9265979)	None	None	None	None	None	
2	(0, 0.052631576)	(1, 0.052631576)	(2, 0.5789474)	(3, 0.052631576)	(4, 0.052631576)	(5, 0.052631576)	(6, 0.052
3	(0, 0.11111111)	(1, 0.11111111)	(2, 0.11111111)	(3, 0.11111111)	(4, 0.11111111)	(5, 0.11111111)	(6, 0.1
4	(0, 0.025643678)	(1, 0.025643671)	(2, 0.6443898)	(3, 0.02564623)	(4, 0.025641352)	(5, 0.17609452)	(6, 0.025
5	(0, 0.11111111)	(1, 0.11111111)	(2, 0.11111111)	(3, 0.11111111)	(4, 0.11111111)	(5, 0.11111111)	(6, 0.1
6	(0, 0.052632395)	(1, 0.052634556)	(2, 0.5789191)	(3, 0.05263158)	(4, 0.05263158)	(5, 0.052648533)	(6, 0.05
7	(0, 0.052645355)	(1, 0.052659817)	(2, 0.052645687)	(3, 0.05263158)	(4, 0.5788612)	(5, 0.05264827)	(6, 0.05
8	(0, 0.016951028)	(1, 0.01695477)	(2, 0.016956108)	(3, 0.016963044)	(4, 0.016949866)	(5, 0.01695069)	(6, 0.8
9	(0, 0.05264412)	(1, 0.052634716)	(2, 0.05264591)	(3, 0.05263595)	(4, 0.05263848)	(5, 0.05263359)	(6, 0.5
10	(0, 0.034488987)	(1, 0.034485165)	(2, 0.72405535)	(3, 0.03448666)	(4, 0.034485813)	(5, 0.03450167)	(6, 0.034
11	(6, 0.93796873)	None	None	None	None	None	

可以看到相关的一些预测数据

LDA模型的预测数据导出到了DocumentTopic.csv文件里。

```
]: doctop.to_csv('DocumentTopic.csv')
```

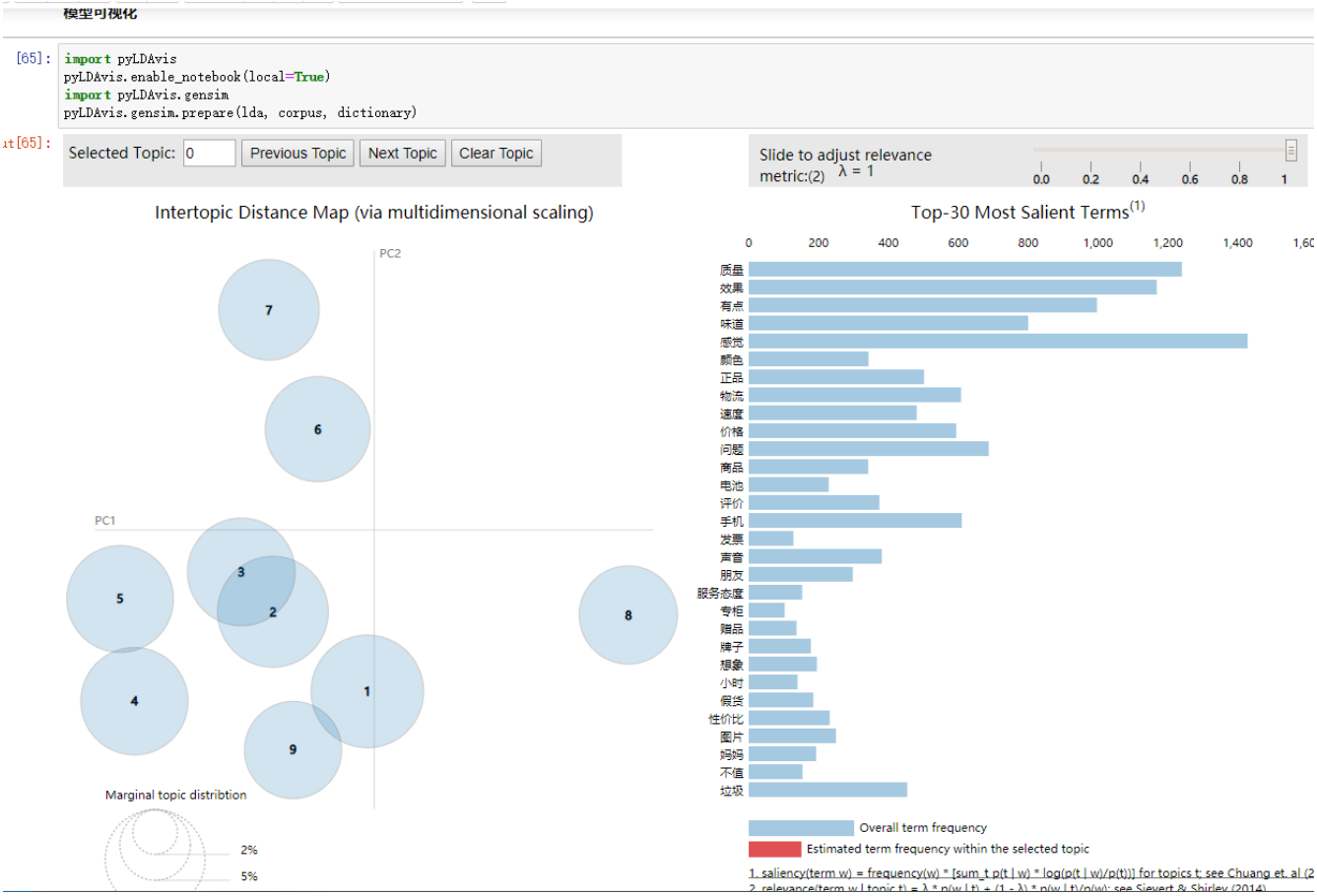
主题对应的关键词

```
]: topicwords = lda.print_topics(num_words=20)
   topicwords
```

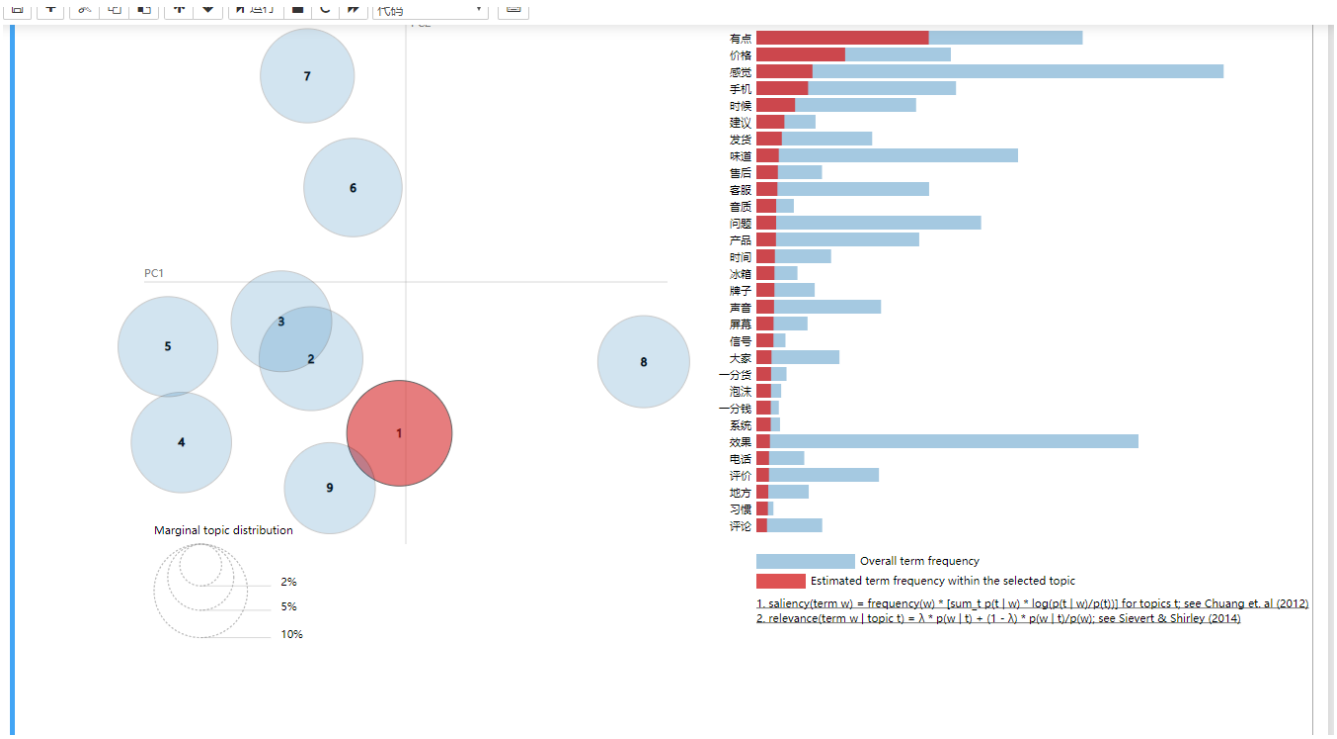
```
]: [(0,
  '0.046*"物流" + 0.041*"速度" + 0.039*"颜色" + 0.019*"电池" + 0.018*"垃圾" + 0.016*"发货" + 0.014*"味道
*"商家" + 0.008*"性价比" + 0.007*"气味" + 0.007*"客服" + 0.007*"感觉" + 0.007*"产品" + 0.006*"价钱" + 0.
 (1,
  '0.074*"有点" + 0.038*"价格" + 0.024*"感觉" + 0.022*"手机" + 0.017*"时候" + 0.012*"建议" + 0.011*"发货
*"产品" + 0.008*"时间" + 0.008*"冰箱" + 0.008*"牌子" + 0.008*"声音" + 0.007*"屏幕" + 0.007*"信号" + 0.00
 (2,
  '0.042*"问题" + 0.020*"质量" + 0.019*"产品" + 0.017*"物流" + 0.016*"商品" + 0.016*"想象" + 0.015*"服务
009*"评论" + 0.009*"垃圾" + 0.009*"电视" + 0.008*"购物" + 0.008*"效果" + 0.007*"味道" + 0.007*"卖家" + 0
 (3,
  '0.019*"有点" + 0.014*"效果" + 0.013*"感觉" + 0.012*"垃圾" + 0.012*"手机" + 0.012*"时候" + 0.011*"问题
*"朋友" + 0.006*"总体" + 0.006*"产品" + 0.005*"温度" + 0.005*"价格便宜" + 0.005*"结果" + 0.004*"尺寸" +
 (4,
  '0.090*"质量" + 0.027*"感觉" + 0.024*"手机" + 0.021*"商品" + 0.014*"大家" + 0.013*"妈妈" + 0.010*"卖家
*"图片" + 0.008*"评价" + 0.007*"物流" + 0.007*"狗狗" + 0.007*"手感" + 0.006*"客服" + 0.006*"问题" + 0.00
 (5,
  '0.077*"效果" + 0.028*"评价" + 0.017*"产品" + 0.017*"速度" + 0.016*"客服" + 0.015*"感觉" + 0.014*"假货
*"皮肤" + 0.009*"卖家" + 0.009*"功能" + 0.007*"问题" + 0.007*"质量" + 0.007*"垃圾" + 0.006*"蚊子" + 0.00
 (6,
  '0.050*"感觉" + 0.044*"效果" + 0.032*"质量" + 0.026*"声音" + 0.020*"有点" + 0.019*"价格" + 0.018*"正品
*"内容" + 0.009*"产品" + 0.009*"外观" + 0.009*"问题" + 0.008*"不值" + 0.007*"朋友" + 0.007*"结果" + 0.00
 (7,
  '0.061*"感觉" + 0.018*"味道" + 0.013*"小时" + 0.013*"专柜" + 0.012*"时间" + 0.012*"手机" + 0.010*"基本
*"有点" + 0.009*"问题" + 0.009*"产品" + 0.009*"效果" + 0.008*"浪费" + 0.008*"垃圾" + 0.007*"孩子" + 0.00
 (8,
  '0.062*"味道" + 0.040*"正品" + 0.029*"感觉" + 0.023*"质量" + 0.020*"朋友" + 0.015*"发票" + 0.013*"客服
*"地方" + 0.010*"头发" + 0.009*"发货" + 0.008*"结果" + 0.008*"衣服" + 0.007*"态度" + 0.007*"划痕" + 0.00
```

主题对应的一些关键词表示。

主题建模的可视化



由图可知，图中共有9个气泡，分散在四个象限中，且重叠区域较少。一个好的主题模型将在整个图表中分散相当大的非重叠气泡，而不是聚集在一个象限中，因此本次主题建模模型良好。

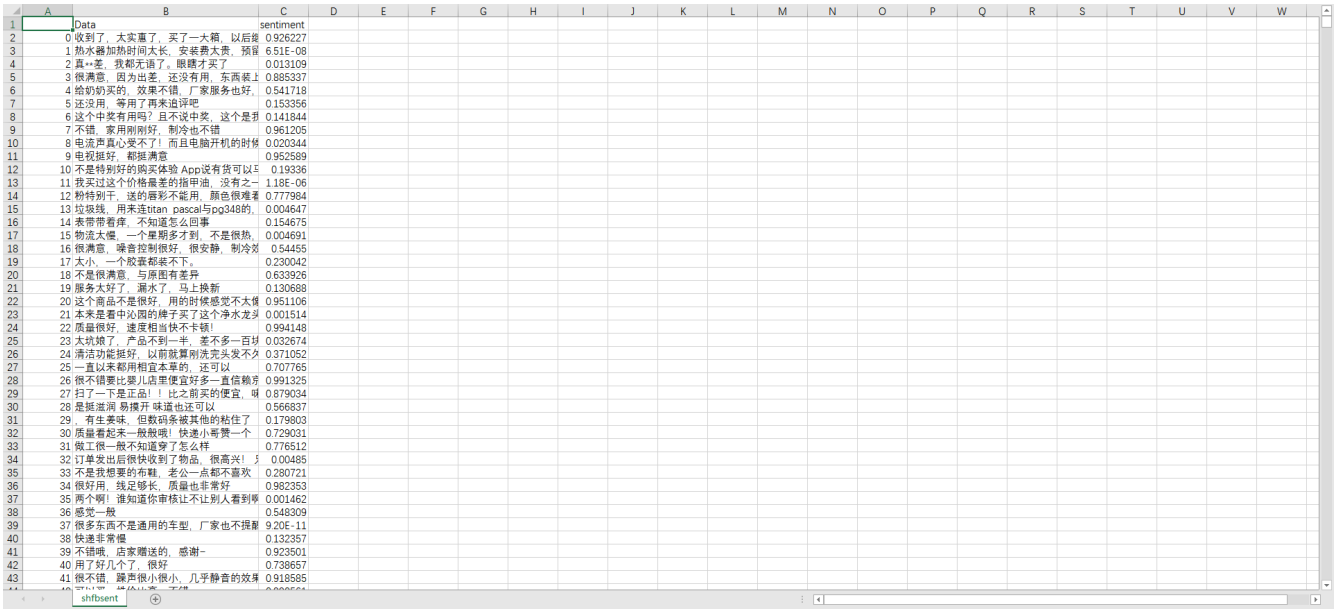


1号主题气泡最大，说明该主题在样本数据中最普遍。将光标移到改气泡上，可以看到该主题的显著关键词为“有点”、“价格”、“感觉”、“手机”、“时候”，其中“有点”的占比最大，其余四个词汇占比递减。因此，该主题可能为“有点”主题。

情感分析结果

中文情感分析

虽然中文情感分析的准确率只有70%，但是本次的情感分析结果还有有一定的参考价值的。 在情感倾向分析中，数值为[0,1]，其中 0.5 是绝对中性，0是消极，1是积极。由下图可以看出，数值大于0.5的数据 量远远大于数值小于0.5的数据量，因此，本次文本挖掘的情感倾向为正面远大于负面，即大多数内容属于正面消息



	A	B	C
1		Data	sentiment
2	0	收到了，太实惠了，买了一大箱，以后	0.926227
3	1	热水器加热时间太长，安装费太贵，预	6.51E-08
4	2	真**差，我都无语了。眼瞎才买了	0.013109
5	3	很满意，因为出差，还没有用，东西装上	0.885337
6	4	给奶奶买的，效果不错，厂家服务也好，	0.541718
7	5	还没用，等用了再来追评吧	0.153356
8	6	这个中奖有用吗？且不说中奖，这个是开	0.141844
9	7	不错，家用刚刚好，制冷也不错	0.961205
10	8	电流声真心受不了！而且电脑开机的时候	0.020344
11	9	电视挺好，都挺满意	0.952599
12	10	不是特别好的购买体验 App说有货可以5	0.19336
13	11	我买过这个价格最差的指甲油，没有之一	1.18E-06
14	12	粉特别干，送的唇彩不能用，颜色很难看	0.777984
15	13	垃圾线，用来连titan pascal与pg348的，	0.004647
16	14	表带带着痒，不知道怎么回事	0.154675
17	15	物流太慢，一个星期多才到，不是很热，	0.004691
18	16	很满意，噪音控制很好，很安静，制冷效	0.54455
19	17	太小，一个般套都装不下。	0.230042
20	18	不是很满意，与原图有差异	0.633926
21	19	服务太好了，漏水了，马上换新	0.130688
22	20	这个商品不是很好，用的时候感觉不太像	0.951106
23	21	本来是看中沁园的牌子买了这个净水龙头	0.001514
24	22	质量很好，速度相当快不卡顿！	0.994148
25	23	太坑娘了，产品不到一半，差不多一百块	0.032674
26	24	清洁功能挺好，以前就算刚洗完头发不久	0.371052
27	25	一直以来都用相宜本草的，还可以	0.707765
28	26	很不错要比婴儿店里便宜好多一直信赖京	0.991325
29	27	扫了一下是正品！！比之前买的便宜，哦	0.879034
30	28	是挺滋润 易摸开 味道也还可以	0.566837
31	29	有生姜味，但数码条被其他的粘住了	0.179803
32	30	质量看起来一般般哦！快递小哥赞一个	0.729031
33	31	做工很一般不知道穿了怎么样	0.776512
34	32	订单发出后很快收到了物品，很高兴！	0.00485
35	33	不是我想要的布鞋，老公一点都不喜欢	0.280721
36	34	很好用，线足够长，质量也非常好	0.982353
37	35	两个啊！谁知道你审核让不让别人看到啊	0.001462
38	36	感觉一般	0.548309
39	37	很多东西不是通用的车型，厂家也不提醒	9.20E-11
40	38	快递非常慢	0.132357
41	39	不错哦，店家赠送的，感谢~	0.923501
42	40	用了好几个了，很好	0.738657
43	41	很不错，踩声很小很小，几乎静音的效果	0.918585

shfbsent

其中越接近1表示情感越偏向正面，越趋于0说明越负面

相关资料导出到了shfbsent.csv里。

根据相关的数据可以看出来最终得到的情感分析数据还是很合理的。

参考链接

基于主题的文本情感分析

结巴分词

TF-IDF

K-Means聚类

主题建模

LDA模型