**­**

**电商产品评论爬取和数据情感分析**

**姓 名：**

**指导教师姓名：**

***注：详细代码和数据见附件***

目录

[一． 网站爬取数据 3](#_Toc1971)

[1.首先打开一个店铺找到评论 3](#_Toc6095)

[2.分析网页 3](#_Toc17932)

[3.继续分析我们需要的东西 3](#_Toc23827)

[4.尝试在css中找寻字体文件 4](#_Toc14964)

[5.找出正确的字体文件 4](#_Toc25568)

[6.将网页数据保存到本地并分析 找到（x,y）和字体的映射关系 5](#_Toc26654)

[7.根据找到的规律 取出字体文件中所有字体 还有位置 存储到字典中 5](#_Toc19026)

[8.现在开始获取所有css文件中的类对应的坐标 5](#_Toc17824)

[9.解密文字 6](#_Toc25057)

[10.获取数据并且保存 6](#_Toc18124)

[二． 综合分析建模题:电商产品评论数据情感分析 7](#_Toc32369)

[1. 背景与挖掘目标 7](#_Toc10755)

[2.案例分析以及基本步骤 7](#_Toc24255)

[3.评论预处理 8](#_Toc19809)

[（1）评论去重 8](#_Toc358)

[（2）数据清洗 8](#_Toc12870)

[4.评论分词 9](#_Toc4813)

[（1）分词，词性标注，去除停用词 9](#_Toc18933)

[（2）提取含名词的评论 11](#_Toc17337)

[（3）绘制词云查看分词效果 14](#_Toc15462)

[5.构建模型 15](#_Toc5477)

[（1）评论数据感情倾向分析 15](#_Toc8836)

[（2）使用LDA模型进行主题分析 21](#_Toc20291)

[6.作品总结 27](#_Toc14181)

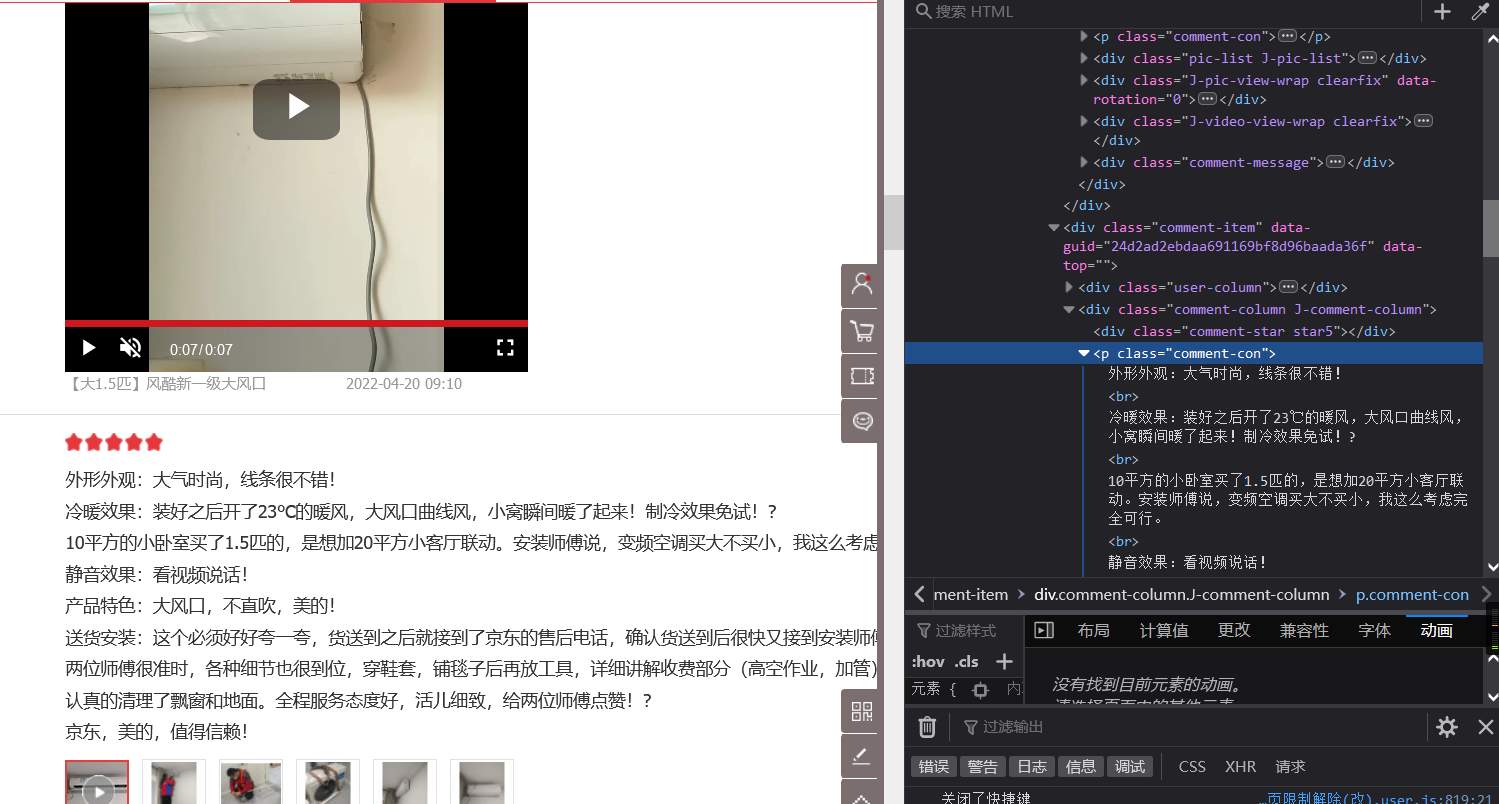
# 网站爬取数据

1.首先打开一个店铺找到评论



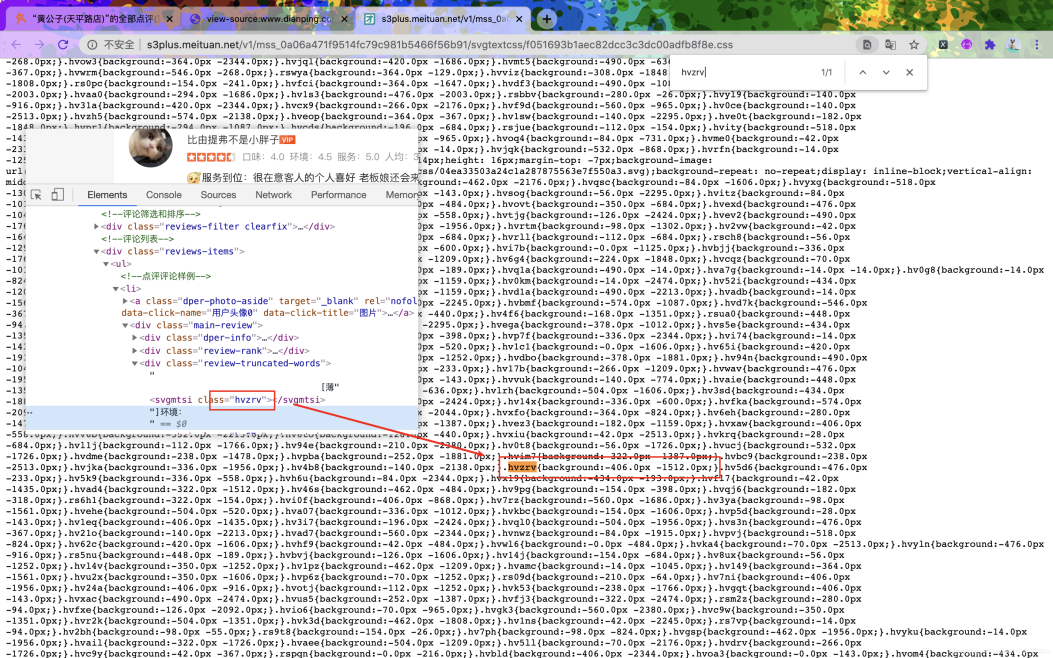
2.分析网页

查看到下面有些字体经过[加密](https://so.csdn.net/so/search?q=%E5%8A%A0%E5%AF%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_44173603/article/details/_blank)处理，刷新页面会发现：每一次加密的字体是不一样的。查看网页源代，查看所有css，发现这个css就是我们想要用的文件。那么现在我们就要用代码来获取到这个css文件的url。



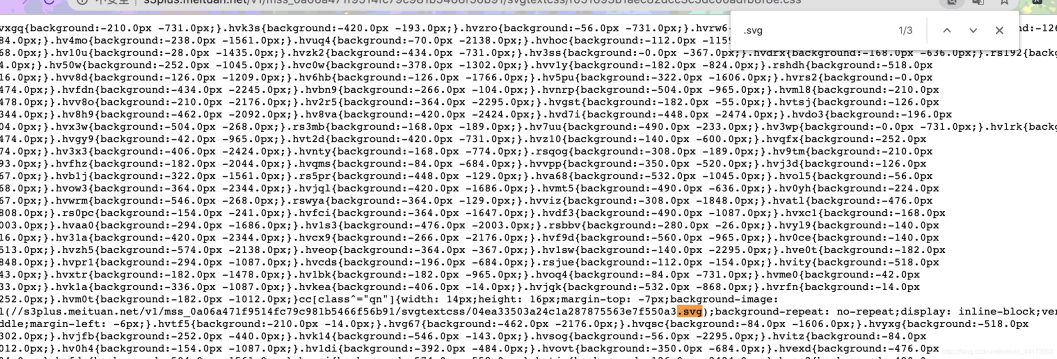
3.继续分析我们需要的东西

打开这个css文件，发现上一个页面加密的字体的类。在这个css文件中可以用查找到，后面有对应的坐标。



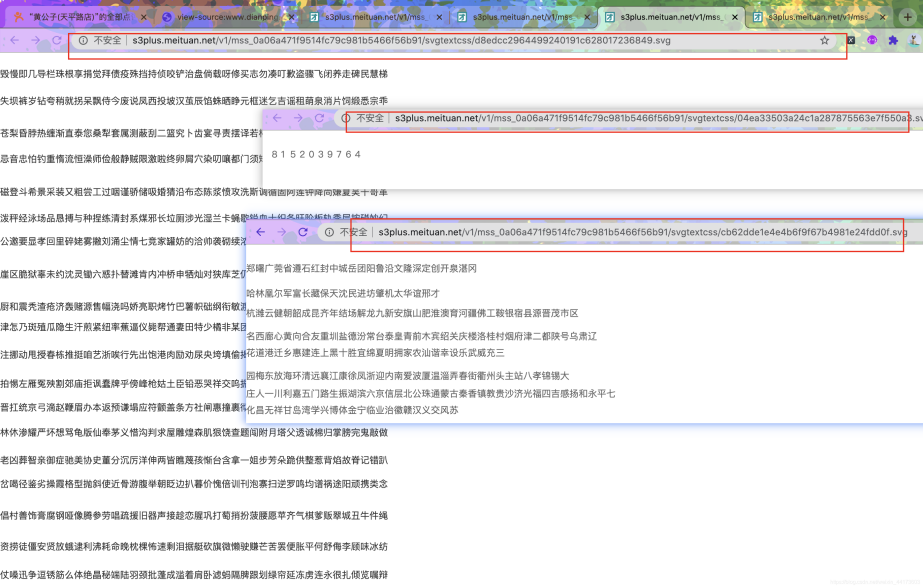
4.尝试在css中找寻字体文件

查询css文件中，有没有我们想要的字体文件 command + f 或者 ctrl  
+f 查询。发现文件中有三个字体文件，分别打开三个文件的url，发现只有一个字体映射文件是正确的。



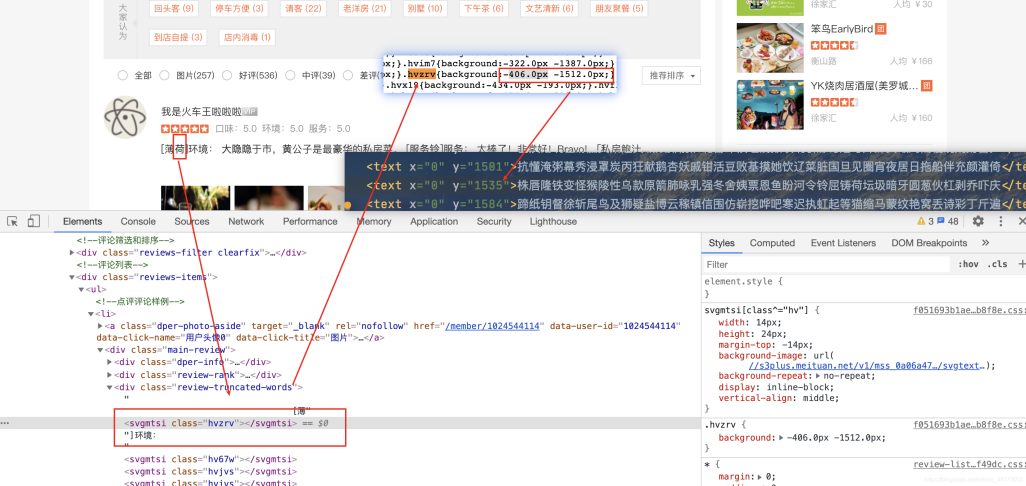
5.找出正确的字体文件

打开三个字体文件的url，发现正确的就是最多的，也就是最大的一个文件。我们不能凭着url来判断哪个字体文件更大，所以要访问。根据返回的数据，来判断正确的文件是哪个。接下来就要访问url来获取字体文件的内容，然后将最大的字体文件内容存储起来，方便替换。



6.将网页数据保存到本地并分析 找到（x,y）和字体的映射关系

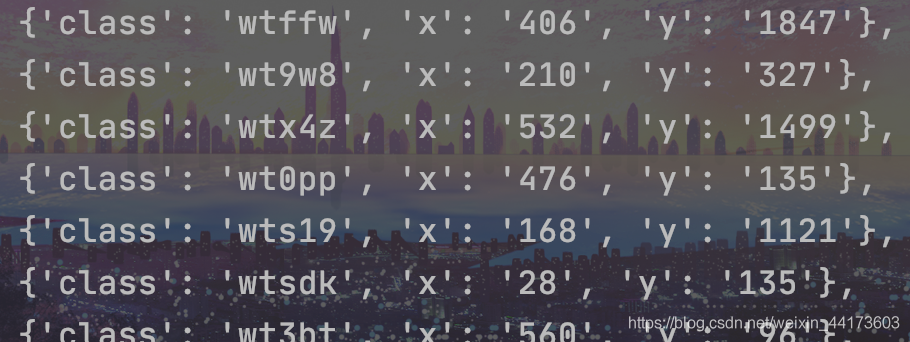
取出字体文件后可以在本地进行手动的查询，摸索文字对应关系。因为每一次刷新都是动态更改css或者svg文件内容，包括每次刷新加密的字都不同。所以将一个保存到本地，根据本地这一个固定的来尝试，尝试成功后动态获取。经过多次尝试，发现规律，两个数字第一个数字除以14。就是文字的下标，此图数字为-406，除以14。下标就是29，第二个数字就在两个y值中间。根据规律 匹配出所有文字的映射关系。



7.根据找到的规律 取出字体文件中所有字体 还有位置 存储到字典中

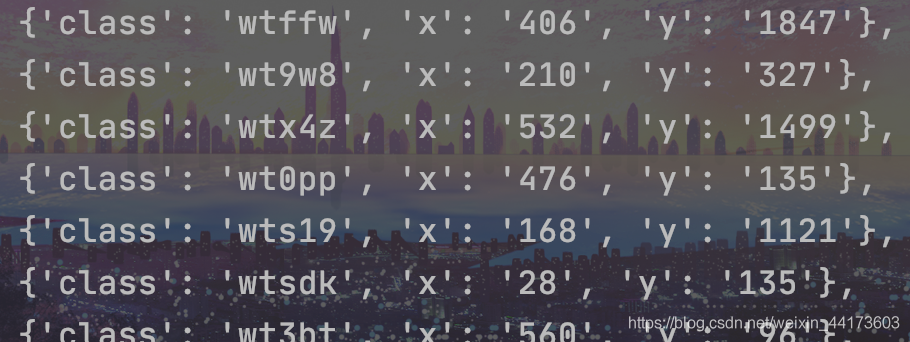
用正则在字体文件中取出数字的x值（即在本行的下标）y值用一个元组来存储，判断时获取加密文字的坐标y值是否在元组两个值中间即可。取出所有的数据 保存到类中的字典。

存储后的字典格式为 value值为字体内容 x为下标 y值为一个元组 用来存储在哪两个数字之间。



8.现在开始获取所有css文件中的类对应的坐标

依旧使用正则来取出所有的数据，查询到的数据，再存储到类中的一个字典。

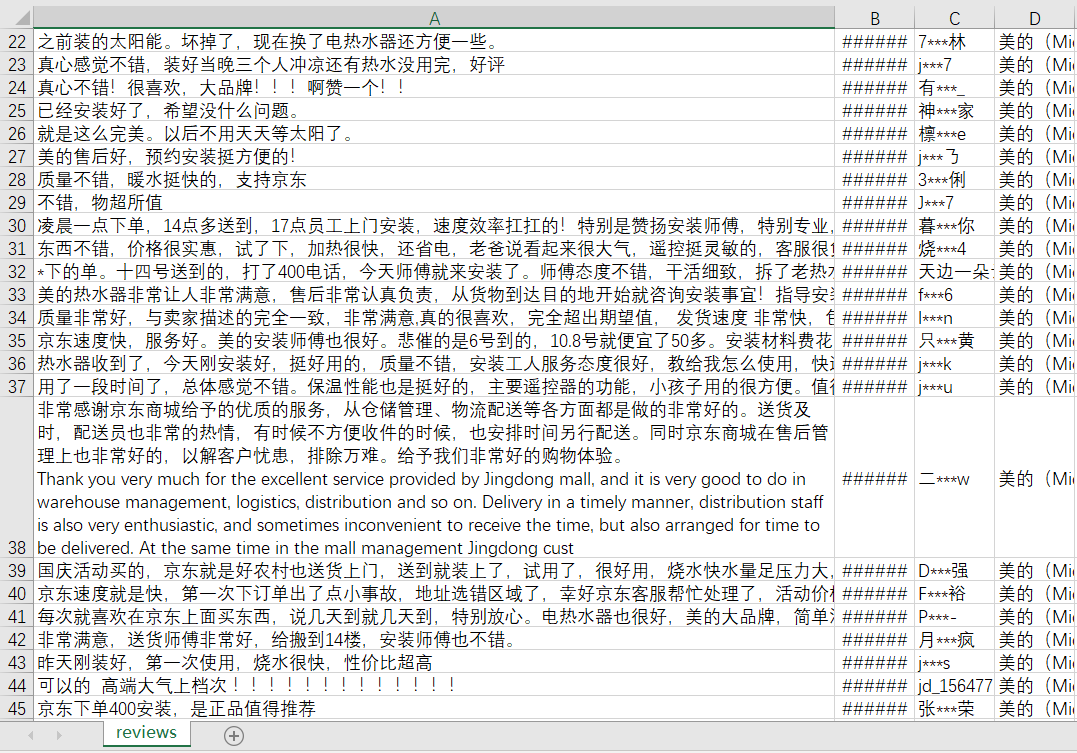


9.解密文字

现在取出网页中被加密的字体的class属性 使用xpath取出就可以。因为现在已经取出了两个字典。一个有加密字体的class属性还有字体的x,y的值，另一个字典中有这个加密字体对应的文字。刚刚我们也取出来所有被加密文字的class属性，只需要循环判断，取出对应的字就可以来。将原来保存的这个网页的text内容中加密的文字，替换为正常的文字。

10.获取数据并且保存

最难的加密已经弄出来了，现在就是一个简单的取数据就可以了。因为长评论和短评论存储的xpath不同，所以需要一个小判断 。运行代码，查看数据即可，控制台打印以及保存到本地的csv文件。



# 综合分析建模题:电商产品评论数据情感分析

1. 背景与挖掘目标

随着电子商务的迅速发展和网络购物的流行，人们对于网络购物的需求变得越来越高，也给电商企业带来巨大的发展机遇，与此同时，这种需求也推动了更多电商企业的崛起，引发了激烈的竞争。而在这种激烈竞争的大背景下，除了提高商品质量、压低价格外，了解更多消费者的心声对电商企业来说也变得越来越有必要。其中非常重要的方式就是对消费者的评论文本数据进行内在信息的分析。

评论信息中蕴含着消费者对特定产品和服务的主观感受，反应了人们的态度、立场和意见，具有非常宝贵的研究价值。一方面，对企业来说，企业需要根据海量的评论文本数据去更好的了解用户的个人喜好，从而提高产品质量，改善服务，获取市场上的竞争优势。另一方面，消费者需要在没有看到真正的产品实体、做出购买决策之前，根据其他购物者的评论了解产品的质量、性价比等信息，为购物抉择提供参考依据。

**需求：**

（1） 对京东商城中美的电热水器的评论进行情感分析；

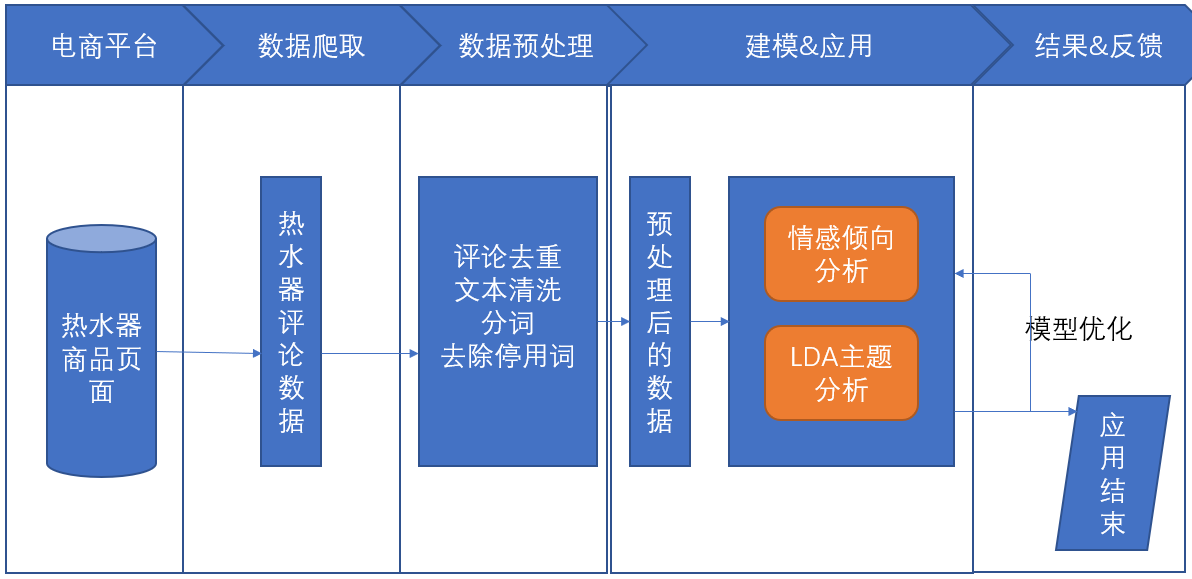
（2） 从评论文本中挖掘出用户的需求、意见，购买原因以及产品的优缺点；

（3） 根据模型结果给出改善产品的建议。

2.案例分析以及基本步骤

电商产品评论数据情感分析流程，主要步骤如下：

1. 利用python对京东商城中美的电热水器的评论数据进行数据清洗，分词，停用词过滤等操作
2. 对预处理后的数据进行情感分析，将评论文本数据按照情感倾向分为正面评论数据（好评）和负面评论数据（差评）
3. 分别对正、负面评论数据进行LDA主题分析，从对应的结果分析文本评论数据中有价值的内容
4. 电商产品评论数据清洗情感分析流程图



3.评论预处理

（1）评论去重

一些电商平台为了避免一些客户长时间不进行评论,往往会设置一道程序，如果用户超过规定的时间仍然没有做出评论，系统就会自动替客户做出评论，这类数据显然没任何分析价值

由语言的特点可知，在大多数情况下，不同购买者之间的有价值的评论是不会出现完全重复的,如果不同购物者的评论完全重复,那么这些评论一般都是毫无意义的。显然这种评论中只有最早的评论才有意义(即只有第一条有作用)。

有的部分评论相似程度极高，可是在某些词语的运用上存在差异，此类评论即可归为重复评论，若是删除文字相近评论,则会出现误删的情况。由于相近的评论也存在不少有用的信息,去除这类评论显然不合适。因此,为了存留更多的有用语料，针对完全重复的语料下手，仅删除完全重复部分,以确保保留有用的文本评论信息。

评论去重代码如下

|  |
| --- |
| #一.评论预处理  #1.评论去重  reviews = pd.read\_csv("数据/reviews.csv")  reviews = reviews[['content','content\_type']].drop\_duplicates()  content = reviews['content'] |

（2）数据清洗

通过观察数据发现，评论中惨杂着许多数字与字母对于本案例的挖掘目标而言，这类数据本身并没有实质性帮助。另外,由干该评论文本数据主要是围绕京东商城中美的电热水器进行评价的,其中“京东”“京东商城”“美的”“热水器”“电热水器”等词出现的频数很大,但是对分析目标并没有什么作用，因此可以在分词之前将这些词去除

数据清洗代码如下：

|  |
| --- |
| #2.数据清洗  strinfo = re.compile('[0-9a-zA-Z]|京东|美的|电热水器|热水器|')  content = content.apply(lambda x: strinfo.sub('', x)) |

4.评论分词

（1）分词，词性标注，去除停用词

1. 对评论数据进行分词

分词是文本信息处理的基础环节，是将一个单词序列切分成单个单词的过程。准确地分词可以极大地提高计算机对文本信息的识别和理解能力。相反，不准确的分词将会产生大量的噪声，严重干扰计算机的识别理解能力，并对这些信息的后续处理工作产生较大的影响。

汉语的基本单位是字，由字可以组成词，由词可以组成句子,进而由一些句子组成段、节、章。篇可风加里需要外理二管由文语料从中正确地识别出词是一件非常基础且重要的工作。

然而,中文以字为基本书写单付词与词之间沿右明显的区分标记。中文分词的任务就是把中文的序列切分成右音的曰印i人迁的词虫估很所形成的词串反映句子

的本意。

例如：我帮小方打饭

输出：我 帮 小方 打饭

使用基于词典的中文分词方法进行中文信息处理时，应该考思未登录问的处理。未登录词是指词典中没有登录过的人名、地名、机构名、译名及新词语等。当采用匹配的办法来切分词语时，由于词典中没有登录这些词，会引起自动切分词语的困难。常见的未登陆词有命名实体，如“张三”“北京”“联想集团”等;专业术语，如“贝叶斯算法”“模态”“万维网”;新词语,如“卡拉OK"“美刀”“啃老族”等。

另外,中文分词还存在切分歧义问题,如“当结合成分子时”这个句子可以有以下切分方法:“当/结合/成分/子时”“当/结合/成/分子/时”“当/结/合成/分子/时”“当/结/合成分/子时”等。

可以说，中文分词的关键问题为切分歧义的消解和未登录词的识别。

词典匹配是分词最为传统也最为常见的一种办法。匹配方式可以为正向（从左到右)或逆向(从右到左)。对于匹配中遇到的多种分段可能性（Segmentation Ambiguity),通常会选取数目最少的词分隔出来。

很明显，这种方式对词表的依赖很大，一旦出现词表中不存在的新词，算法是无法做到正确切分的。但是词表匹配也有它的优势，比如简单易懂、不依赖训练数据、易于纠错等。

还有一类方法是通过语料数据中的统计特征（如互信号量）去估计相邻汉字之间的关联性，进而实现词的切分。

分词最常用的工作包是jieba分词包，jieba包是python写成的一个分词开源库，专门用来中文分词。

1. 去除停用词

停用词(Stop Words)词曲译为“电脑检索中的虚字，非检索用字”。在SEO搜索引擎中，为节省存储空间和提高搜索效率,搜索引擎在索引页面或处理搜索请求时会自动忽略某些字或词，这些字或词即被称为停用词。

停用词一定程度上相当于过滤词(Filter Words),区别是过滤词的范围更大一些,包含色情、政治等敏感信息的关键词都会被视做过滤词加以处理,停用词本身则没有这个限制。通常意义上说,停用词大致可分为以下两类:一类是使用使用十分广泛，甚至过于频繁的一些单词，一类是文本中出现频率很高，但实际意义不大的词

分词，词性标注，去除停用词代码如下：

|  |
| --- |
| #二.评论分词  #分词，词性标注，去除停用词代码  #分词  worker = lambda s: [(x.word,x.flag) for x in psg.cut(s)] #自定义简单分词函数  seg\_word = content.apply(worker)  #将词语转为数据框形式，一列是词，一列是词语所在的句子id，最后一列是词语在该句子中的位置  n\_word = seg\_word.apply(lambda x:len(x)) #每一评论中词的个数  n\_content = [[x+1]\*y for x,y in zip(list(seg\_word.index),list(n\_word))]  index\_content = sum(n\_content,[]) #将嵌套的列表展开，作为词所在评论的id  seg\_word = sum(seg\_word,[])  word = [x[0] for x in seg\_word] #词  nature = [x[1] for x in seg\_word] #词性  content\_type = [[x]\*y for x,y in zip(list(reviews['content\_type']),  list(n\_word))]  content\_type = sum(content\_type,[]) #评论类型  result = pd.DataFrame({"index\_content":index\_content,  "word":word,  "nature":nature,  "content\_type":content\_type})  #删除标点符号  result = result[result['nature'] != 'x'] #x为标点符号  #删除停用词  stop\_path = open("数据/stoplist.txt",'r',encoding='utf-8')  stop = stop\_path.readlines()  stop = [x.replace('\n','') for x in stop]  word = list(set(word)-set(stop))  result = result[result['word'].isin(word)]  #构造各词在对应评论的位置列  n\_word = list(result.groupby(by = ['index\_content'])['index\_content'].count())  index\_word = [list(np.arange(0,y)) for y in n\_word]  index\_word = sum(index\_word,[]) #表示词语在该评论中的位置  #合并评论id，评论中词的id，词，词性，评论类型  result['index\_word'] = index\_word |

经过分词后，评论由一个字符串形式变为多个由文字或词语组成的字符串的形式，可判断评论词中词语是否为停用词，根据停用词的定义整理出停用词词库，并根据词库去除评论中的停用词。

（2）提取含名词的评论

由于本案例的目标是对产品特征的优缺占进行分析，类似“不错,很好的产品”“很不错，继续支持”等评论虽然表达了对产品的情感倾向,但是实际上无法根据这些评论提取出哪些产品特征是用户满意的。评论中只有出现明确的名词，如机构团体及其他专有名词时，才有意义，因此需要对分词后的词语进行词性标注。之后再根据词性将含有名词类的评论提取出来。

jieba关于词典词性标记,采用ICTCLAS 的标记方法。ICTCLAS汉语词性标注集如下

Ag 形语素 形容词性语素。形容词代码为a，语素代码ｇ前面置以A。  
  
a 形容词 取英语形容词adjective的第1个字母。  
  
ad 副形词 直接作状语的形容词。形容词代码a和副词代码d并在一起。  
  
an 名形词 具有名词功能的形容词。形容词代码a和名词代码n并在一起。  
  
b 区别词 取汉字“别”的声母。  
  
c 连词 取英语连词conjunction的第1个字母。  
  
Dg 副语素 副词性语素。副词代码为d，语素代码ｇ前面置以D。  
  
d 副词 取adverb的第2个字母，因其第1个字母已用于形容词。  
  
e 叹词 取英语叹词exclamation的第1个字母。  
  
f 方位词 取汉字“方”  
  
g 语素 绝大多数语素都能作为合成词的“词根”，取汉字“根”的声母。  
  
h 前接成分 取英语head的第1个字母。  
  
i 成语 取英语成语idiom的第1个字母。  
  
j 简称略语 取汉字“简”的声母。  
  
k 后接成分  
  
l 习用语 习用语尚未成为成语，有点“临时性”，取“临”的声母。  
  
m 数词 取英语numeral的第3个字母，n，u已有他用。  
  
Ng 名语素 名词性语素。名词代码为n，语素代码ｇ前面置以N。  
  
n 名词 取英语名词noun的第1个字母。  
  
nr 人名 名词代码n和“人(ren)”的声母并在一起。  
  
ns 地名 名词代码n和处所词代码s并在一起。  
  
nt 机构团体 “团”的声母为t，名词代码n和t并在一起。  
  
nz 其他专名 “专”的声母的第1个字母为z，名词代码n和z并在一起。  
  
o 拟声词 取英语拟声词onomatopoeia的第1个字母。  
  
p 介词 取英语介词prepositional的第1个字母。  
  
q 量词 取英语quantit的第1个字母。  
  
r 代词 取英语代词pronoun的第2个字母,因p已用于介词。  
  
s 处所词 取英语space的第1个字母。  
  
Tg 时语素 时间词性语素。时间词代码为t,在语素的代码g前面置以T。  
  
t 时间词 取英语time的第1个字母。  
  
u 助词 取英语助词auxiliary  
  
Vg 动语素 动词性语素。动词代码为v。在语素的代码g前面置以V。  
  
v 动词 取英语动词verb的第一个字母。  
  
vd 副动词 直接作状语的动词。动词和副词的代码并在一起。  
  
vn 名动词 指具有名词功能的动词。动词和名词的代码并在一起。  
  
w 标点符号  
  
x 非语素字 非语素字只是一个符号，字母x通常用于代表未知数、符号。  
  
y 语气词 取汉字“语”的声母。  
  
z 状态词 取汉字“状”的声母的前一个字母。  
-------------------------------------------------------------------------------  
a:        形容词  
b:        区别词  
c:        连词  
d:        副词  
e:        叹词  
g:        语素字  
h:        前接成分  
i:        习用语  
j:        简称  
k:        后接成分  
m:        数词  
n:        普通名词  
nd:        方位名词  
nh:        人名  
ni:        机构名  
nl:        处所名词  
ns:        地名  
nt:        时间词  
nz:        其他专名  
o:        拟声词  
p:        介词  
q:        量词  
r:        代词  
u:        助词  
v:        动词  
wp:        标点符号  
ws:        字符串  
x:        非语素字

提取含义“n”的评论代码如下：

|  |
| --- |
| #提取含有名词的评论  # 提取含有名词类的评论  ind = result[['n' in x for x in result['nature']]]['index\_content'].unique()  result = result[[x in ind for x in result['index\_content']]] |

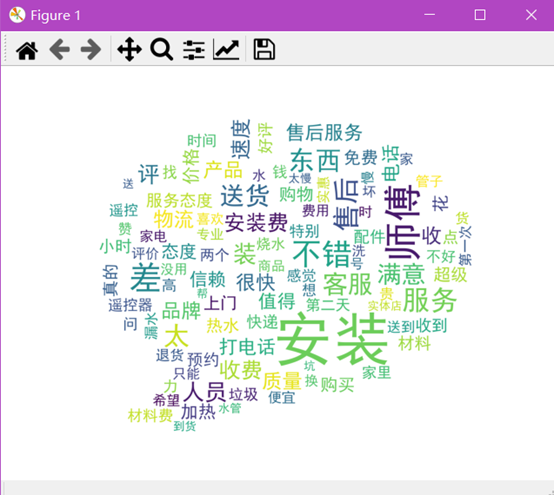
（3）绘制词云查看分词效果

预处理后，通过绘制云图来查看分词效果，词云会将出现频率高的“关键词”更加突出，首先将词语进行词频统计，统计出词频前100的词语，利用wordcloud模块的WordCloud绘制词云。

绘制词云代码如下：

|  |
| --- |
| #绘制词云  import matplotlib.pyplot as plt  from wordcloud import WordCloud  frequencies = result.groupby(by = ['word'])['word'].count()  frequencies = frequencies.sort\_values(ascending = False)  backgroud\_Image=plt.imread('数据/pl.jpg')  wordcloud = WordCloud(font\_path="simhei.ttf",  max\_words=100,  background\_color='white',  mask=backgroud\_Image)  my\_wordcloud = wordcloud.fit\_words(frequencies)  plt.imshow(my\_wordcloud)  plt.axis('off')  plt.show()  # 将结果写出  result.to\_csv("数据/word.csv", index = False, encoding = 'utf-8') |

分词后的词云图



可以看出其中“安装”，“师傅”，“售后”等词出现频率较高。因此可以初步判断用户对产品的这几个方面比较重视。

5.构建模型

（1）评论数据感情倾向分析

1. 匹配情感词

情感倾向又称为情感极佳，在某件商品评论中，用户对该商品表达自身观点，支持，反对还是对立。

首先要对情感词进行匹配，主要采用词典匹配的方法

一般基于词表的情感分析方法，分析的效果往往与情感词表内的词语有较强的相关性，如果情感词表内的词语足够全面，并且词语符合该案例场景下所表达的情感，那么情感分析的效果会更好。针对本案例场景，需要在知网提供的词表基础上进行优化,例如“好评”“超值”“差评”“五分”等词只有在网络购物评论上出现，就可以根据词语的情感倾向添加至对应的情感词表内。将“满意"“好评”“很快”“还好”“还行”“超值”“给力”“支持”“超好”“感谢”“太棒了”“厉害”“挺舒服”“辛苦”“完美”“喜欢”“值得”“省心”等词添加进正面情感词表。将“差评”“贵”“高”“漏水”等词加入负面情感词表。

读入正负面评论情感词表，正面词语赋予初始权重1，负面词语赋予初始权重-1。使用merge函数按照词语情感词表与分词结果进行匹配。

匹配感情词代码如下：

|  |
| --- |
| #匹配情感词  import pandas as pd  import numpy as np  word = pd.read\_csv("数据/word.csv")  # 读入正面、负面情感评价词  pos\_comment = pd.read\_csv("数据/正面评价词语（中文）.txt", header=None,sep="\n",  encoding = 'utf-8', engine='python')  neg\_comment = pd.read\_csv("数据/负面评价词语（中文）.txt", header=None,sep="\n",  encoding = 'utf-8', engine='python')  pos\_emotion = pd.read\_csv("数据/正面情感词语（中文）.txt", header=None,sep="\n",  encoding = 'utf-8', engine='python')  neg\_emotion = pd.read\_csv("数据/负面情感词语（中文）.txt", header=None,sep="\n",  encoding = 'utf-8', engine='python')  # 合并情感词与评价词  positive = set(pos\_comment.iloc[:,0])|set(pos\_emotion.iloc[:,0])  negative = set(neg\_comment.iloc[:,0])|set(neg\_emotion.iloc[:,0])  intersection = positive&negative # 正负面情感词表中相同的词语  positive = list(positive - intersection)  negative = list(negative - intersection)  positive = pd.DataFrame({"word":positive,  "weight":[1]\*len(positive)})  negative = pd.DataFrame({"word":negative,  "weight":[-1]\*len(negative)})  posneg = positive.append(negative)  # 将分词结果与正负面情感词表合并，定位情感词  data\_posneg = posneg.merge(word, left\_on = 'word', right\_on = 'word',  how = 'right')  data\_posneg = data\_posneg.sort\_values(by = ['index\_content','index\_word']) |

1. 修改感情倾向

情感倾向修正主要根据情感词前而两个位置的词语具否在在否定词而去判断情感值的正确与否,由于汉语中存在多重否定现象,即当否定词出现奇数次时,表示否定意思;当否定词出现偶数次时，表示肯定意思。按照汉语习惯,搜索每个情感词前两个词语,若出现奇数否定词,则调整为相反的情感极性。

本案例使用的否定词表共有19个否定词，分别为:不、没、无、非、莫、弗、毋、未、否、别、燕、休、不是、不能、不可、没有、不用、不要、从没、不太。

读入否定词表,对情感值的方向进行修正。计算每条评论的情感得分,将评论分为正面评论和负面评论,并计算情感分析的准确率

修改情感倾向代码如下：

|  |
| --- |
| #修正情感倾向  # 根据情感词前时候有否定词或双层否定词对情感值进行修正  # 载入否定词表  notdict = pd.read\_csv("数据/not.csv")  # 处理否定修饰词  data\_posneg['amend\_weight'] = data\_posneg['weight'] # 构造新列，作为经过否定词修正后的情感值  data\_posneg['id'] = np.arange(0, len(data\_posneg))  only\_inclination = data\_posneg.dropna() # 只保留有情感值的词语  only\_inclination.index = np.arange(0, len(only\_inclination))  index = only\_inclination['id']  for i in np.arange(0, len(only\_inclination)):  review = data\_posneg[data\_posneg['index\_content'] ==  only\_inclination['index\_content'][i]] # 提取第i个情感词所在的评论  review.index = np.arange(0, len(review))  affective = only\_inclination['index\_word'][i] # 第i个情感值在该文档的位置  if affective == 1:  ne = sum([i in notdict['term'] for i in review['word'][affective - 1]])  if ne == 1:  data\_posneg['amend\_weight'][index[i]] = -\  data\_posneg['weight'][index[i]]  elif affective > 1:  ne = sum([i in notdict['term'] for i in review['word'][[affective - 1,  affective - 2]]])  if ne == 1:  data\_posneg['amend\_weight'][index[i]] = -\  data\_posneg['weight'][index[i]]    # 更新只保留情感值的数据  only\_inclination = only\_inclination.dropna()  # 计算每条评论的情感值  emotional\_value = only\_inclination.groupby(['index\_content'],  as\_index=False)['amend\_weight'].sum()  # 去除情感值为0的评论  emotional\_value = emotional\_value[emotional\_value['amend\_weight'] != 0] |

1. 查看情感分析结果

使用Wordcloud函数分别对正面评论和负面评论绘制词云，以查看情感分析效果

查看情感分析结果代码如下：

|  |
| --- |
| #查看情感分析效果  # 给情感值大于0的赋予评论类型（content\_type）为pos,小于0的为neg  emotional\_value['a\_type'] = ''  emotional\_value['a\_type'].loc[emotional\_value['amend\_weight'] > 0] = 'pos'  emotional\_value['a\_type'].loc[emotional\_value['amend\_weight'] < 0] = 'neg'  # 查看情感分析结果  result = emotional\_value.merge(word,  left\_on = 'index\_content',  right\_on = 'index\_content',  how = 'left')  result = result[['index\_content','content\_type', 'a\_type']].drop\_duplicates()  confusion\_matrix = pd.crosstab(result['content\_type'], result['a\_type'],  margins=True) # 制作交叉表  (confusion\_matrix.iat[0,0] + confusion\_matrix.iat[1,1])/confusion\_matrix.iat[2,2]  # 提取正负面评论信息  ind\_pos = list(emotional\_value.loc[emotional\_value['a\_type'] == 'pos']['index\_content'])  ind\_neg = list(emotional\_value.loc[emotional\_value['a\_type'] == 'neg']['index\_content'])  posdata = word[[i in ind\_pos for i in word['index\_content']]]  negdata = word[[i in ind\_neg for i in word['index\_content']]]  # 绘制词云  import matplotlib.pyplot as plt  from wordcloud import WordCloud  # 正面情感词词云  freq\_pos = posdata.groupby(by = ['word'])['word'].count()  freq\_pos = freq\_pos.sort\_values(ascending = False)  backgroud\_Image=plt.imread('数据/pl.jpg')  wordcloud = WordCloud(font\_path="simhei.ttf",  max\_words=100,  background\_color='white',  mask=backgroud\_Image)  pos\_wordcloud = wordcloud.fit\_words(freq\_pos)  plt.imshow(pos\_wordcloud)  plt.axis('off')  plt.show()  # 负面情感词词云  freq\_neg = negdata.groupby(by = ['word'])['word'].count()  freq\_neg = freq\_neg.sort\_values(ascending = False)  neg\_wordcloud = wordcloud.fit\_words(freq\_neg)  plt.imshow(neg\_wordcloud)  plt.axis('off')  plt.show()  # 将结果写出,每条评论作为一行  posdata.to\_csv("数据/posdata.csv", index = False, encoding = 'utf-8')  negdata.to\_csv("数据/negdata.csv", index = False, encoding = 'utf-8') |

正面情感评论词云



由正面情感词云可知，“不错”，“满意”，“好评”等正面情感词出现频次较高，并且没有负面情感词语。

负面情感评论词云



由负面情感词云可知，“差评”，“垃圾”，“不好”等负面情感词出现频次较高，并且没有正面情感词语。

为了进一步查看分词效果，假定用户在评论时不存在“选了好标签而写了差评内容”的情况，比较原评论的评论类型与情感分析得出评论类型，绘制情感倾向分析混淆矩阵。

混淆矩阵如表

|  |  |  |
| --- | --- | --- |
|  | neg | Pos |
| neg | 363 | 55 |
| Pos | 40 | 443 |

通过比较原评论的评论类型与情感分析得出的评论类型，基于词表的情感分析的准确率达到了89.46%，证明通过词表的感情分析来判断某文本的情感程度是有效地。

（2）使用LDA模型进行主题分析

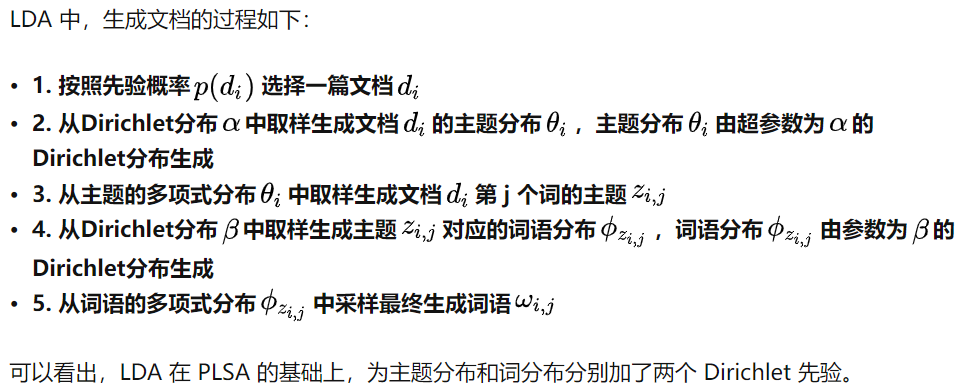
1. LDA主题模型

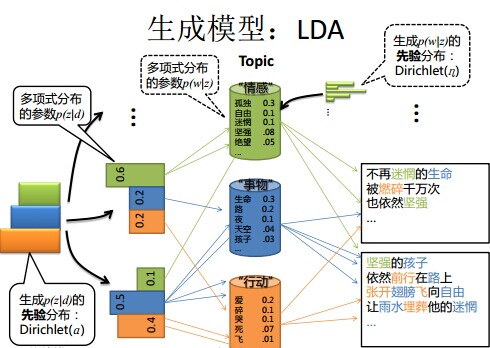
在机器学习领域，LDA是两个常用模型的简称：Linear Discriminant Analysis 和 Latent Dirichlet Allocation。本文的LDA仅指代Latent Dirichlet Allocation. LDA 在主题模型中占有非常重要的地位，常用来文本分类。

LDA由Blei, David M.、Ng, Andrew Y.、Jordan于2003年提出，用来推测文档的主题分布。它可以将文档集中每篇文档的主题以概率分布的形式给出，从而通过分析一些文档抽取出它们的主题分布后，便可以根据主题分布进行主题聚类或文本分类。

**词袋模型**

LDA 采用词袋模型。所谓词袋模型，是将一篇文档，我们仅考虑一个词汇是否出现，而不考虑其出现的顺序。在词袋模型中，“我喜欢你”和“你喜欢我”是等价的。与词袋模型相反的一个模型是n-gram，n-gram考虑了词汇出现的先后顺序。





LDA主题模刑是一种无监督的模式,只需要提供训练文档,就可以自动训练出各种概率，无须任何人工标注讨程、节省了大量的人力及时间。它在文本聚类、主题分析、相似度计算等方面都有广泛的应用。相对于其他主题模型,其引入了狄利克雷先验知识。因此,模型的泛化能力较强,不易出现过拟合现象。

LDA主题模型可以解决多种指代问题,例如,在对美的电热水器的评论中，根据分词的一般规则，经过分词的语句会将“费用”一词单独分割出来，而“费用”是指安装费用还是热水器费用等其他情况?如果只是简单地进行词频统计及情感分析，是无法识别的。这种指代不明的问题并不能准确地反应用户情况,运用 LDA主题模型则可以求得词汇在主题中的概率分布，进而判断“费用”一词属于哪个主题，并求得属于这一主题的概率和同一主题下的其他特征词，从而解决多种指代问题。

建立LDA模型首先要建立词典和语料库

建立词典及语料库代码如下：

|  |
| --- |
| #建立词典及语料库  import pandas as pd  import numpy as np  import re  import itertools  import matplotlib.pyplot as plt  # 载入情感分析后的数据  posdata = pd.read\_csv("数据/posdata.csv", encoding = 'utf-8')  negdata = pd.read\_csv("数据/negdata.csv", encoding = 'utf-8')  from gensim import corpora, models  # 建立词典  pos\_dict = corpora.Dictionary([[i] for i in posdata['word']]) # 正面  neg\_dict = corpora.Dictionary([[i] for i in negdata['word']]) # 负面  # 建立语料库  pos\_corpus = [pos\_dict.doc2bow(j) for j in [[i] for i in posdata['word']]] # 正面  neg\_corpus = [neg\_dict.doc2bow(j) for j in [[i] for i in negdata['word']]] # 负面 |

1. 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，用相对较少的迭代找到最优的主题结构。

具体步骤如下：

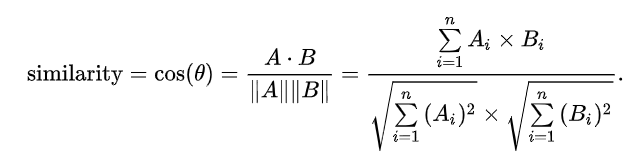
1)取初始主题数k值,得到初始植刑,计算各主题之间的相似度(平均余弦距离)。

2)增加或减少k值,重新训练模刑,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E量主斯间的相似程街,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(A1,A2,…,An),B是(B1,B2,…,Bn)，则A与B的夹角0的余弦值通过以下式计算。

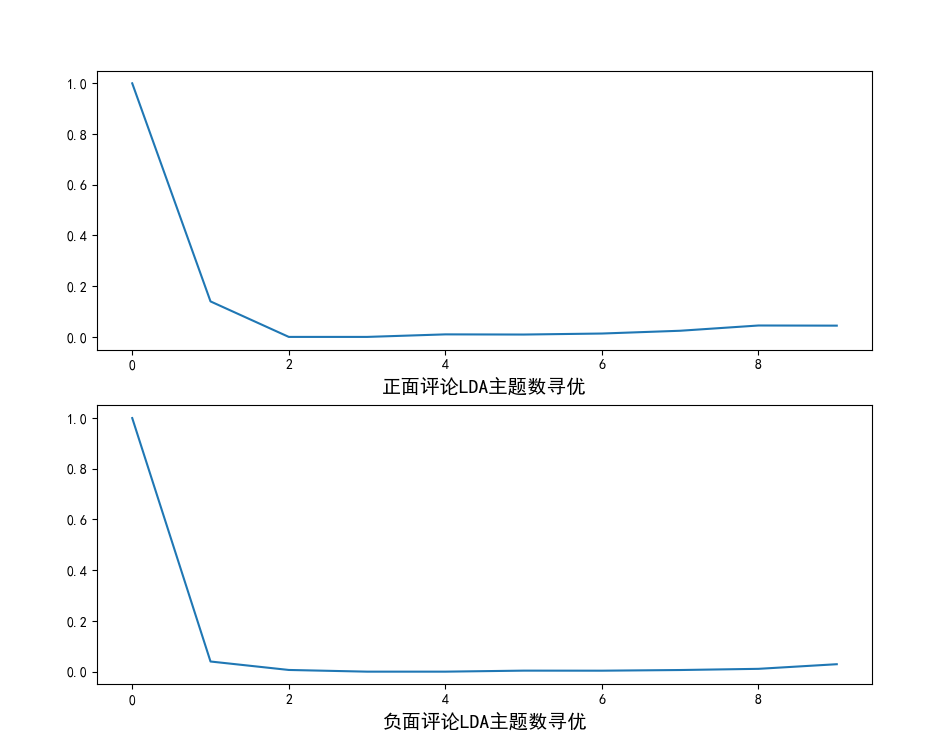


使用LDA主题模型，找出不同主题数下的主题词,每个模型各取出若干个主题词(比如前100个),合并成一个集合。生成任何两个主题间的词频向量,计算两个向量的余弦相似度,值越大就表示越相似;计算各个主题数的平均余弦相似度，寻找最优主题数。

主题数寻优代码如下：

|  |
| --- |
| # 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 = 50)    # 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)    #计算主题平均余弦相似度  pos\_k = lda\_k(pos\_corpus, pos\_dict)  neg\_k = lda\_k(neg\_corpus, neg\_dict)  #绘制主题平均余弦相似度图形  from matplotlib.font\_manager import FontProperties  font = FontProperties(size=14)  #解决中文显示问题  plt.rcParams['font.sans-serif']=['SimHei']  plt.rcParams['axes.unicode\_minus'] = False  fig = plt.figure(figsize=(10,8))  ax1 = fig.add\_subplot(211)  ax1.plot(pos\_k)  ax1.set\_xlabel('正面评论LDA主题数寻优', fontproperties=font)  ax2 = fig.add\_subplot(212)  ax2.plot(neg\_k)  ax2.set\_xlabel('负面评论LDA主题数寻优', fontproperties=font) |

可得主题间平均余弦相似度图如下：



由图可知,对于正面评论数据，当主题数为2或3时，主题间的平均余弦相似度就达到了最低。因此,对正面评论数据做 LDA,可以选择主题数为3;对于负面评论数据，当主题数为3时，主题间的平均余弦相似度也达到了最低。因此,对负面评论数据做LDA，也可以选择主题数为3。

1. 评价主题分析结果

根据主题数寻优结果,使用Python的Gensim模块对正面评论数据和负面评论数据分别构建LDA主题模型,设置主题数为3，经过LDA主题分析后，每个主题下生成10个最有可能出现的词语以及相应的概率

LDA主题分析

|  |
| --- |
| #LDA主题分析  pos\_lda = models.LdaModel(pos\_corpus, num\_topics = 3, id2word = pos\_dict)  neg\_lda = models.LdaModel(neg\_corpus, num\_topics = 3, id2word = neg\_dict)  pos\_lda.print\_topics(num\_words = 10)  neg\_lda.print\_topics(num\_words = 10) |

**美的电热水器正面评价潜在主题如下：**

[(0, '0.121\*"安装" + 0.019\*"客服" + 0.017\*"信赖" + 0.016\*"人员" + 0.015\*"物流" + 0.012\*"品牌" + 0.012\*"速度" + 0.010\*"收" + 0.010\*"第二天" + 0.009\*"小时"'),

(1, '0.064\*"满意" + 0.030\*"很快" + 0.029\*"不错" + 0.020\*"售后" + 0.017\*"太" + 0.014\*"装" + 0.012\*"态度" + 0.012\*"赞" + 0.012\*"服务态度" + 0.011\*"快递"'),

(2, '0.042\*"师傅" + 0.031\*"送货" + 0.027\*"服务" + 0.024\*"值得" + 0.022\*"好评" + 0.019\*"东西" + 0.015\*"购物" + 0.015\*"差" + 0.015\*"送" + 0.013\*"家里"')]

以上数据反映了美的电热水器**正面**评价文本中的潜在主题，**主题1**中的高频特征词,关注点主要是**客服、信赖、快速**等，主要反映美的电热水器**客服服务好值得信赖**等;**主题2**中的高频特征词,即关注点主要是**很快、售后，快递**等，主要反映美的电热水器的**发货速度快、售后服务态度好**等;**主题3**中的高频特征词，即**师傅、好评、值得**等，主要反映京东美的电热水器**安装师傅服务好**。

**美的电热水器负面评价潜在主题如下：**

[(0,

'0.032\*"垃圾" + 0.031\*"师傅" + 0.017\*"不好" + 0.016\*"贵" + 0.016\*"慢" + 0.013\*"人员" + 0.012\*"坑人" + 0.012\*"真的" + 0.012\*"材料" + 0.011\*"坑"'),

(1,

'0.032\*"售后" + 0.020\*"装" + 0.019\*"收" + 0.018\*"打电话" + 0.016\*"太慢" + 0.012\*"问" + 0.011\*"上门" + 0.010\*"评" + 0.010\*"钱" + 0.008\*"两个"'),

(2,

'0.132\*"安装" + 0.030\*"太" + 0.028\*"差" + 0.025\*"安装费" + 0.022\*"东西" + 0.020\*"客服" + 0.019\*"小时" + 0.018\*"收费" + 0.016\*"加热" + 0.016\*"烧水"')]

以上数据反映了美的电热水器**负面**评价文本中的潜在主题，**主题1**中的高频特征词,关注点主要是**贵、慢、材料**等，主要反映美的电热水器**安装师傅收费高，安装慢，材料不好**等;**主题2**中的高频特征词,即关注点主要是**太慢、售后，上门，打电话**等，主要反映美的电热水器的**发货速度慢、售后服务差**等;**主题3**中的高频特征词，即**安装费、加热、客服**等，主要反映京东美的电热水器**客服服务差，电热水器加热性能存在问题，收费过高等**。

6.作品总结

通过综合以上对主题及其中的高频特征词的分析得出美的由热水器有价格实惠、性价比高、外观好看、服务好等优势。相对而言，用户对美的由热水器的抱怨点主要体现在安装的费用高及售后服务差等方面。

因此,用户的购买原因可以总结为以下几个方面,美的是大品牌值得信赖、美的电热水器价格实惠、性价比高。

根据对京东平台上美的电热水器的用户评价情况进行LDA主题模型分析,对美的品牌提出以下几点建议:

①在保持热水器使用方便、价格实惠等优点的基础上，对热水器进行加热功能上的改进,从整体上提升热水器的质量。

②提升安装人员及客服人员的整体素质,提高服务质量,注重售后服务。建立安装

费用收取的明文细则,并进行公布，以减少安装过程中乱收费的现象。适度降低安装费用和材料费用,以此在大品牌的竞争中凸显优势。