**数据挖掘案例实战：利用LDA主题模型提取京东评论数据**

网上购物已经成为大众生活的重要组成部分。人们在电商平台上浏览商品和购物，产生了海量的用户行为数据，其中用户对商品的评论数据对商家具有重要的意义。利用好这些碎片化、非结构化的数据，将有利于企业在电商平台上的持续发展，对这部分数据进行分析，依据评论数据来优化现有产品也是大数据在企业经营中的实际应用。

本章主要针对用户在电商平台上留下的评论数据，对其进行分词、词性标注和去除停用词等文本预处理。基于预处理后的数据进行情感分析，并使用LDA主题模型提取评论关键信息，了解用户的需求、意见、购买原因，以及产品的优缺点，最终提出改善产品的建议。

**一、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，具有非常宝贵的研宄价值。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。

请根据提供的数据实现以下目标。

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。

## 二、分析方法与过程

图1为电商产品评论数据情感分析流程，主要步骤如下。

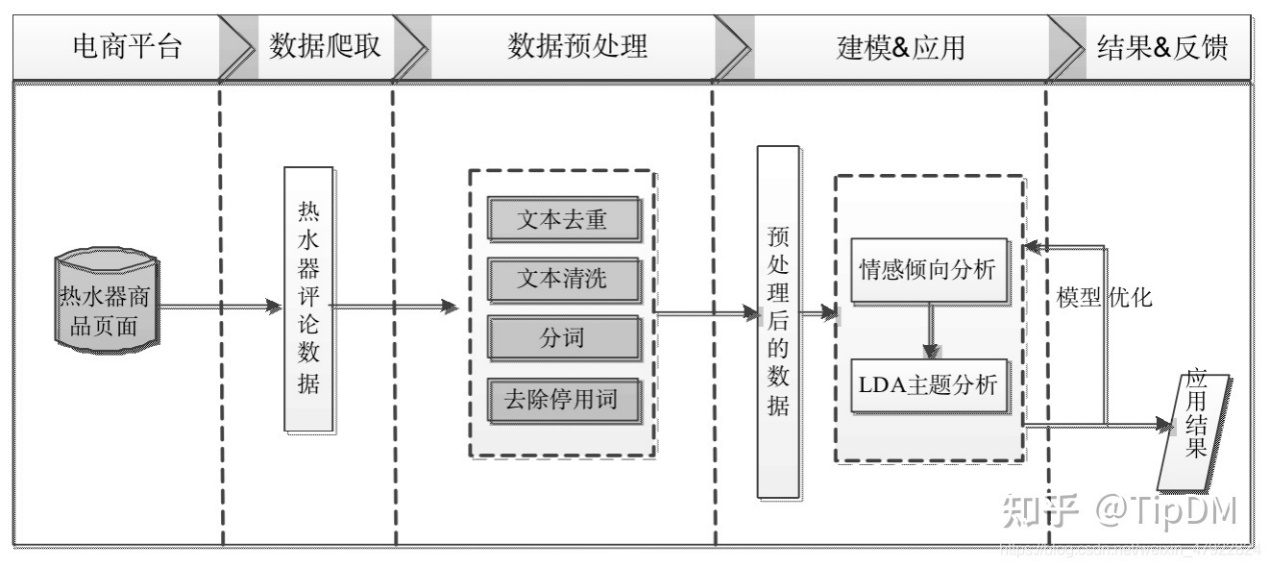
1. 利用Python对京东商城中美的电热水器的评论进行爬取。
2. 利用Python爬取到的京东商城中美的电热水器的评论数据，对评论文本数据进行数据清洗、分词、停用词过滤等操作。
3. 对预处理后的数据进行情感分析，将评论文本数据按照情感倾向分为正面评论数据（好评）和负面评论数据（差评）。
4. 分别对正、负面评价数据进行LDA主题分析，从对应的结果分析文本评论数据中有价值的内容。

图 1 电商产品评论数据情感分析流程

### 1.评论预处理

对京东商城中美的热水器评论数据进行预处理前，需要先对评论数据进行采集。本案例利用Python网络爬虫技术对京东商城中美的热水器评论数据进行采集。由于本案例的重点是对电商产品评论数据情感分析，且网络数据的爬取具有时效性，因此，本案例不再详细介绍数据的采集过程。以下分析所使用的数据与分析结果，仅作为范例参考。

### 2. 评论去重

一些电商平台往往为了避免一些客户长时间不进行评论，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，系统会自动替客户做出评论，这类数据显然没有任何分析价值。

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都不会出现完全重复，如果出现了不同购物者的评论完全重复，这些评论一般都是毫无意义的。这种评论显然只有最早的评论才有意义（即只有第一条有作用）。

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，本节针对完全重复的语料下手，仅删除完全重复部分，以确保尽可能保留有用的文本评论信息。评论去重的代码如代码清单1所示。

代码清单1 评论去重的代码

import pandas as pd

import re

import jieba.posseg as psg

import numpy as np

# 去重，去除完全重复的数据

reviews = pd.read\_csv("../tmp/reviews.csv")

reviews = reviews[['content', 'content\_type']].drop\_duplicates()

content = reviews['content']

运行代码可知，美的热水器的评论共2000条，经过文本去重，共删除重复评论26条，剩余评论1974条。

### 3. 数据清洗

通过人工观察数据发现，评论中夹杂许多数字与字母，对于本案例挖掘目标而言，这类数据本身没有实质性帮助。另外，由于该评论文本数据主要围绕京东商城中美的电热水器进行评价，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，对数据进行清洗，如代码清单2所示。

代码清单2 数据清洗

# 去除去除英文、数字等

# 由于评论主要为京东美的电热水器的评论，因此去除这些词语

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

**分词、词性标注、去除停用词。**

**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一个一个单词的过程。准确的分词可以极大的提高计算机对文本信息的是被和理解能力。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。可见，如果需要处理一篇中文语料，从中正确的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位，词与词之间没有明显的区分标记。中文分词的任务就是把中文的序列切分成有意义的词，即添加合适的词串使得所形成的词串反映句子的本意，中午分词例子如表1所示。

表1中文分词例子

|  |  |
| --- | --- |
| 操作 | 内容 |
| 输入 | 我帮小明打饭 |
| 输出 | 我 帮 小明 打饭 |

当使用基于词典的中文分词方法进行中文信息处理时不得不考虑未登录词的处理。未登录词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”“酒井法子”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”。

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），通常会选取分隔出来词的数目最少的。

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的切分的。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，易于纠错等等。

还有一类方法是通过语料数据中的一些统计特征（如互信息量）去估计相邻汉字之间的关联性，进而实现词的切分。这类方法不依赖词表，特别是在对生词的发掘方面具有较强的灵活性，但是也经常会有精度方面的问题。

分词最常用的工作包是jieba分词包，jieba分词是python写成的一个分词开源库，专门用于中文分词，其有三条基本原理，即实现所采用技术。

1. 基于trie树结构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）。jieba分词自带了一个叫做dict.txt的词典，里面有2万多条词，包含了词条出现的次数（这个次数是作者自己基于人民日报语料等资源训练得出来的）和词性。trie树是有名的前缀树，若一个词语的前面几个字一样，表示该词语具有相同的前缀，可以使用trie树来存储，trie树存储方式具有查找速度快的优势。后一句的“生成句子中汉字所有可能成词情况所构成的有向无环图”意思是给定一个待切分的句子，生成一个如图2所示的有向无环图。

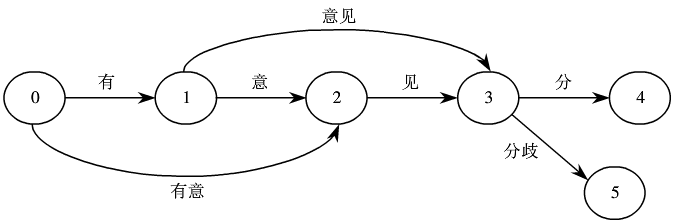


图2“有意见分歧”切分生成的有向无环图

1. 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。先查找待分词句子中已经切分好的词语，再查找该词语出现的频率，然后根据动态规划查找最大概率路径的方法，对句子从右往左反向计算最大概率（反向是因为汉语句子的重心经常落在右边，从右往左计算，正确率要高于从左往右计算，这个类似于逆向最大匹配），最后得到最大概率的切分组合。
2. 对于未登录词，采用HMM模型，使用了Viterbi算法，将中文词汇按照BEMS四个状态来标记。其中B是begin，表示开始位置；E是end，表示结束位置；M是middle，表示中间位置；S是singgle，表示单独成词的位置。HMM模型采用(B,E,M,S)这四种状态来标记中文词语，比如北京可以标注为BE，即北/B京/E，表示北是开始位置，京是结束位置，中华民族可以标注为BMME，就是开始、中间、中间和结束。

**去除停用词**

停用词（Stop Words），词典译为“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，包含情色、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，停用词大致可分为如下两类。

一类是使用十分广泛，甚至是过于频繁的一些单词。比如英文的“i”“is”“what”，中文的“我”“就”等，这些词几乎在每个文档上均会出现，查询这样的词无法保证搜索引擎能够给出真正相关的搜索结果，因此无法缩小搜索范围来提高搜索结果的准确性，同时还会降低搜索的效率。因此，在搜索的时候，Google和百度等搜索引擎会忽略掉特定的常用词，如果使用了太多的停用词，有可能无法得到精确的结果，甚至可能得到大量毫不相关的搜索结果。

另一类是文本中出现频率很高，但实际意义又不大的词。这一类词主要包括了语气助词、副词、介词、连词等，通常自身并无明确意义，只有将其放入一个完整的句子中才有一定作用的词语。常见的有“的”“在”“和”“接着”等，例如“泰迪教育研究院是最好的大数据知识传播机构之一”这句话中的“是”“的”就是两个停用词。

经过分词后，评论由一个字符串的形式变为多个由文字或词语组成的字符串的形式，可判断评论中词语是否为停用词。根据上述停用词的定义整理出停用词库，并根据停用词库去除评论中的停用词，如代码清单3所示。

**代码清单3 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表示标点符号**

**# 删除停用词**

**stop\_path = open("../data/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, []) # 表示词语在改评论的位置**

**# 合并评论id，评论中词的id，词，词性，评论类型**

**result['index\_word'] = index\_word**

**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点进行分析，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，如机构团体及其它专有名词时，评论才有意义，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汉语词性标注集如表2所示。

表2 ICTCLAS 汉语词性标注集

|  |
| --- |
|  |
| 代码 | 名称 | 帮助记忆的诠释 | 代码 | 名称 | 帮助记忆的诠释 |
| 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 | ns | 地名 | 名词代码n和处所词代码s并在一起 |
| a | 形容词 | 取英语形容词adjective的第1个字母 | nt | 机构团体 | “团”的声母为t，名词代码n和t并在一起 |
| 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 | 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 |
| 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 | o | 拟声词 | 取英语拟声词onomatopoeia的第1个字母 |
| b | 区别词 | 取汉字“别”的声母 | p | 介词 | 取英语介词prepositional的第1个字母 |
| c | 连词 | 取英语连词conjunction的第1个字母 | q | 量词 | 取英语quantity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 | r | 代词 | 取英语代词pronoun的第2个字母，因p已用于介词 |
| d | 副词 | 取adverb的第2个字母，因其第1个字母已用于形容词 | s | 处所词 | 取英语space的第1个字母 |
| e | 叹词 | 取英语叹词exclamation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 |
| f | 方位词 | 取汉字“方”的声母 | t | 时间词 | 取英语time的第1个字母 |
| 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 | u | 助词 | 取英语助词auxiliary 的第2个字母，因a已用于形容词 |
| h | 前接成分 | 取英语head的第1个字母 | 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 |
| i | 成语 | 取英语成语idiom的第1个字母 | v | 动词 | 取英语动词verb的第1个字母 |
| j | 简称略语 | 取汉字“简”的声母 | vd | 副动词 | 直接作状语的动词。动词和副词的代码并在一起 |
| k | 后接成分 |  | 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 |
| l | 习用语 | 习用语尚未成为成语，具有“临时性”，取“临”的声母 | w | 标点符号 |  |
| m | 数词 | 取英语numeral的第3个字母，n、u已有他用 | 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 |
| 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 | y | 语气词 | 取汉字“语”的声母 |
| n | 名词 | 取英语名词noun的第1个字母 | z | 状态词 | 取汉字“状”的声母的前一个字母 |
| nr | 人名 | 名词代码n和“人（ren）”的声母并在一起 |  |  |  |

根据得出的词性，提取评论中词性含有“n”的评论，如代码清单4所示。

代码清单4 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']]]

**绘制词云查看分词效果**

进行数据预处理后，可绘制词云查看分词效果，词云会将文本中出现频率较高的“关键词”予以视觉上的突出。首先需要对词语进行词频统计，将词频按照降序排序，选择前100个词，使用wordcloud模块中的WordCloud绘制词云，查看分词效果，如代码清单5所示。

代码清单5 绘制词云

**mport** 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('../data/pl.jpg')

wordcloud **=** WordCloud(font\_path**=**"STZHONGS.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("../tmp/word.csv", index **=** False, encoding **=** 'utf-8')

运行代码清单5可得到分词后的词云图，如图3所示。



图3 分词后的词云图

**根据图3可以看出，对评论数据进行预处理后，分词效果较为符合预期。其中“安装”“师傅”“售后”“物流”，“服务”等词出现频率较高，因此可以初步判断用户对产品这几个方面比较重视。**

**构建模型**

**评论数据情感倾向分析**

**匹配情感词**

情感倾向也称为情感极性。在某商品评论中，可以理解为用户对该商品表达自身观点所持的态度是支持、反对还是中立，即通常所指的正面情感、负面情感、中性情感。由于本案例主要是对产品的优缺点进行分析，因此只要确定用户评论信息中的情感倾向方向分析即可，不需要分析每一评论的情感程度。

对评论情感倾向进行分析首先对情感词进行匹配，主要采用词典匹配的方法，本案例使用的情感词表是2007年10月22日知网发布“情感分析用词语集（beta版）”，主要使用“中文正面评价”词表、“中文负面评价”“中文正面情感”“中文负面情感”词表。将“中文正面评论”“中文正面情感”两个词表合并，并给每个词语赋予初始权重1，作为本案例的正面评论情感词表。将“中文负面评价”“中文负面情感”两个词表合并，并给每个词语赋予初始权重-1，作为本案例的负面评论情感词表。

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函数将按照词语将情感词表与分词结果进行匹配，如代码清单6所示。

代码清单6 匹配情感词

**import** pandas **as** pd

**import** numpy **as** np

word **=** pd**.**read\_csv("../tmp/word.csv")

*# 读入正面、负面情感评价词*

pos\_comment **=** pd**.**read\_csv("../data/正面评价词语（中文）.txt", header**=**None,sep**=**"\n",

encoding **=** 'utf-8', engine**=**'python')

neg\_comment **=** pd**.**read\_csv("../data/负面评价词语（中文）.txt", header**=**None,sep**=**"\n",

encoding **=** 'utf-8', engine**=**'python')

pos\_emotion **=** pd**.**read\_csv("../data/正面情感词语（中文）.txt", header**=**None,sep**=**"\n",

encoding **=** 'utf-8', engine**=**'python')

neg\_emotion **=** pd**.**read\_csv("../data/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'])

**修正情感倾向**

情感方向修正主要根据情感词前面2个位置的词语是否存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2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、無、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，如代码清单7所示。

代码清单7 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("../data/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]

**查看情感分析效果**

使用wordcloud包下的WordCloud函数分别对正面评论和负面评论绘制词云，查看情感分析效果，如代码清单8所示。

代码清单8 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'][emotional\_value['amend\_weight'] **>** 0] **=** 'pos'

emotional\_value['a\_type'][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[emotional\_value['a\_type'] **==** 'pos']['index\_content'])

ind\_neg **=** list(emotional\_value[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('../data/pl.jpg')

wordcloud **=** WordCloud(font\_path**=**"STZHONGS.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("../tmp/posdata.csv", index**=**False, encoding**=**'utf-8')

negdata**.**to\_csv("../tmp/negdata.csv", index**=**False, encoding**=**'utf-8')

运行代码清单8，可得正面情感评论词云如图4所示，负面情感评论词云如图5所示。



图4 正面情感评论词云

由图4正面情感评论词云可知，“不错”“满意”“好评”等正面情感词出现的频数较高，并且没有掺杂负面情感的词语，可以看出情感分析能较好的将正面情感评论抽取出来。



图5 负面情感评论词云

由图5负面情感评论词云可知，“差评”“垃圾”“不好”“太差”等负面情感词出现的频数较高，并且没有掺杂正面情感的词语，可以看出情感分析能较好的将负面情感评论抽取出来。

为了进一步查看情感分析效果，假定用户在评论时，不存在选了好评的标签，而写了差评内容的情况下，比较原评论的评论类型与情感分析得出的评论类型，绘制情感倾向分析混淆矩阵，如表3所示，查看词表的情感分析的准确率。

表3 情感倾向分析混淆矩阵

|  |  |  |
| --- | --- | --- |
|  | neg | pos |
| neg | 363 | 55 |
| pos | 40 | 443 |

由表3可知，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%，证明通过词表的情感分析去判断某文本的情感程度是有效的。

**使用LDA模型进行主题分析**

**了解LDA主题模型**

**① 主题模型介绍**

主题模型在自然语言处理等领域是用来在一系列文档中发现抽象主题的一种统计模型。传统判断两个文档相似性的方法是通过查看两个文档共同出现的单词的多少，如TF（词频）、TF-IDF（词频-逆向文档频率）等，这种方法没有考虑到文字背后的语义关联，例如在两个文档共同出现的单词很少甚至没有，但两个文档是相似的，因此在判断文档相似性时，需要使用主题模型进行语义分析并判断文档相似性。

如果一篇文档有多个主题，则一些特定的可代表不同主题的词语会反复的出现，此时，运用主题模型，能够发现文本中使用词语的规律，并且把规律相似的文本联系到一起，以寻求非结构化的文本集中的有用信息。例如热水器的商品评论文本数据，代表热水器特征的词语如“安装”“出水量”“服务”等会频繁地出现在评论中，运用主题模型，把热水器代表性特征相关的情感描述性词语与应的特征词语联系起来，从而深入了解用户对热水器的关注点及用户对于某一特征的情感倾向。

**② LDA主题模型**

潜在狄利克雷分配，即LDA模型（Latent Dirichlet Allocation，LDA）是由Blei等人在2003年提出的生成式主题模型。生成模型，即认为每一篇文档的每一个词都是通过“一定的概率选择了某个主题，并从这个主题中以一定的概率选择了某个词语”。LDA模型也被称为三层贝叶斯概率模型，包含文档（d）、主题（z）、词（w）三层结构，能够有效对文本进行建模，和传统的空间向量模型（VSM）相比，增加了概率的信息。通过LDA主题模型，能够挖掘数据集中的潜在主题，进而分析数据集的集中关注点及其相关特征词。

LDA模型采用词袋模型（Bag Of Words，BOW）将每一篇文档视为一个词频向量，从而将文本信息转化为易于建模的数字信息。

定义词表大小为L，一个L维向量(1,0,0,…,0,0)表示一个词。由N个词构成的评论记为 [公式] 。假设某一商品的评论集D由M篇评论构成，记为 [公式] 。M篇评论分布着K个主题，记为 [公式] 。记a和b为狄利克雷函数的先验参数，q为主题在文档中的多项分布的参数，其服从超参数为a的Dirichlet先验分布，f为词在主题中的多项分布的参数，其服从超参数b的Dirichlet先验分布。LDA模型图如图6所示。

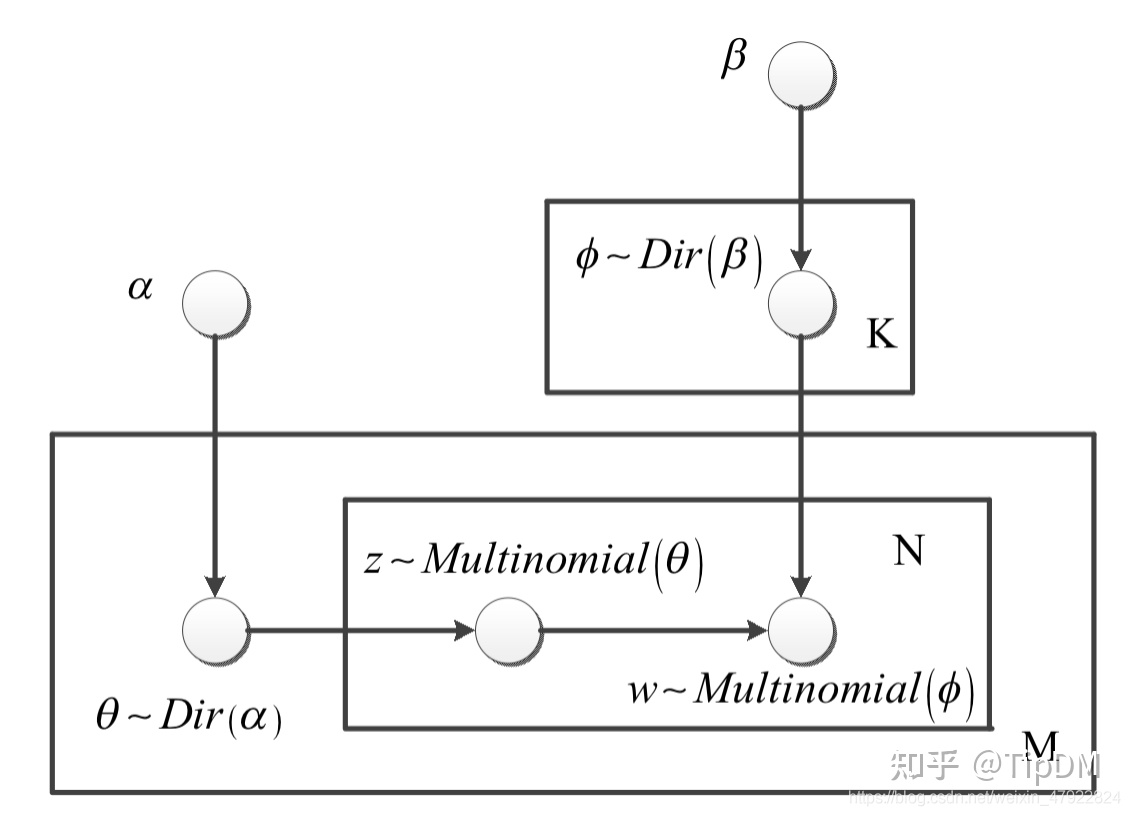


图6 LDA模型结构示意图

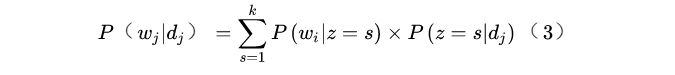
LDA模型假定每篇评论由各个主题按一定比例随机混合而成，混合比例服从多项分布，记为式（1）。

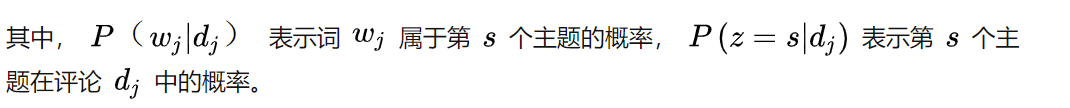


而每个主题由词汇表中的各个词语按一定比例混合而成，混合比例也服从多项分布，记为式（2）。



在评论 [公式] 条件下生成词 [公式] 的概率表示为式（3）。





其中，  表示词  属于第  个主题的概率，  表示第  个主题在评论  中的概率。

LDA主题模型是一种无监督的模式，只需要提供训练文档，它就可以自动训练出各种概率，无需任何人工标注过程，节省大量人力及时间。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，如果简单的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不能购准确的反应用户情况，运用LDA主题模型，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主题模型，首先需要建立词典及语料库，如代码清单9所示。

代码清单9 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("../data/posdata.csv", encoding**=**'utf-8')

negdata **=** pd**.**read\_csv("../data/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*

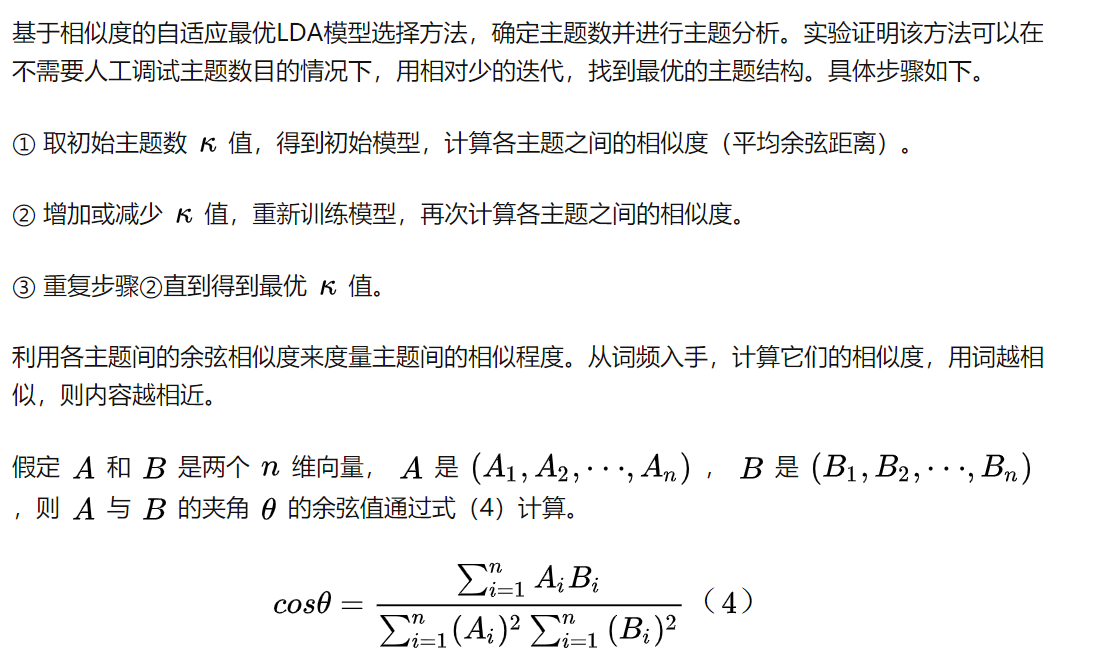
neg\_dict **=** corpora**.**Dictionary([[i] **for** i **in** negdata['word']]) *# 负面*

*# 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*

**寻找最优主题数**



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，如代码清单10所示。

代码清单10 主题数寻优

*# 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)

运行代码清单10，可得主题间平均余弦相似度图，如图7所示。

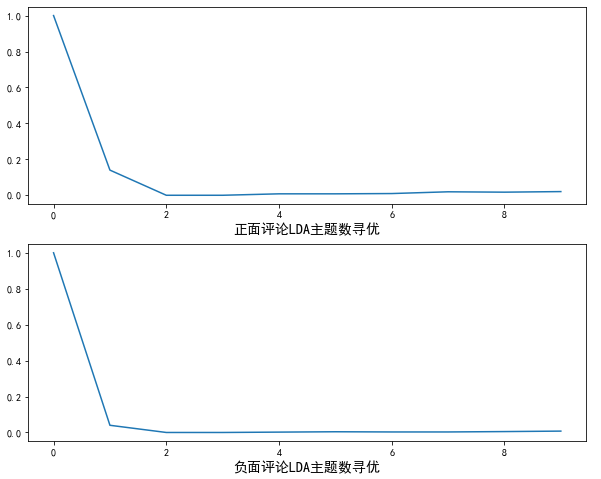


图7 主题间平均余弦相似度图

由图7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就达到了最低。因此，对负面评论数据做LDA，可以选择主题数为3。

**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、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，如代码清单11所示。

代码清单11 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)

运行代码清单11，可得LDA主题分析结果如表4与表5所示。

表4 美的正面评价潜在主题

|  |  |  |
| --- | --- | --- |
| Topic 1 | Topic 2 | Topic 3 |
| 满意 | 值得 | 安装 |
| 师傅 | 太 | 很快 |
| 送货 | 速度 | 不错 |
| 服务 | 家里 | 信赖 |
| 好评 | 电话 | 东西 |
| 客服 | 态度 | 物流 |
| 售后 | 赞 | 购物 |
| 人员 | 收 | 送 |
| 差 | 收到 | 品牌 |
| 质量 | 服务态度 | 装 |

表4反映了美的正面评价文本中的潜在主题，主题1中的高频特征词，即关注点主要是师傅、不错、售后服务等，主要反映美的安装师傅服务好等；主题2中的高频特征词，即关注点主要是物流、价格等，主要反映热水器的发货速度快，及品牌价格实惠等；主题3中的高频特征词，即不错、满意、质量、好评等，主要反映京东美的产品质量不错。

表5 美的负面评价潜在主题

|  |  |  |
| --- | --- | --- |
| Topic 1 | Topic 2 | Topic 3 |
| 安装 | 垃圾 | 师傅 |
| 差 | 售后 | 太 |
| 安装费 | 人员 | 东西 |
| 装 | 配件 | 客服 |
| 不好 | 服务 | 小时 |
| 贵 | 价格 | 收 |
| 烧水 | 送货 | 收费 |
| 真的 | 只能 | 打电话 |
| 坑 | 遥控器 | 加热 |
| 产品 | 速度 | 慢 |

表5反映了美的负面评价文本中的潜在主题，主题1中的高频特征词主要关注点在安装、安装费、收费这几方面，可能存在安装师傅收费过高等问题；主题2中的高频特征词主要与售后、服务这几方面，反映该产品售后服务差等问题；主题3中的高频特征词主要与加热功能有关，即主要反映的是美的热水器加热性能存在问题。

综合以上对主题及其中的高频特征词分析得出，美的热水器的优势有以下几个方面：价格实惠、性价比高、外观好看、服务好。相对而言，用户对美的热水器的抱怨点主要体现在美的热水器安装的费用高及售后服务差等。

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，美的热水器价格实惠，性价比高。

根据对京东平台上，美的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2点建议。

① 在保持热水器使用方便、价格实惠等优点基础上，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，更加注重售后服务。建立安装费用收取明文细则，并进行公开透明，减少安装过程的乱收费问题。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。

**小结**

本案例向读者展示了如何使用Python处理电商文本评论数据。通过使用Python爬取的案例数据，对文本数据进行预处理、分词、去除停用词等操作，在知网情感词表上进行优化，进行基于词表的情感分析，最后使用LDA对正负面评论进行主题分析。从分析某一热水器的用户情感倾向出发，挖掘出该热水器的优点与不足，从而提升对应商品的生产厂家自身的竞争力。

**拓展思考**

AHP（Analytical Hierachy Process，应用层次分析法）是匹兹堡大学T. L. Saaty教授在20世纪70年代初期提出对定性问题进行定量分析的一种渐变灵活的多准则决策方案，其特点在于把复杂问题中的各种因素通过划分为相互联系的有序层次，使之条理化，根据对有一定客观现实的主观两两比较，把专家意见和分析者的客观判断结果直接有效的结合起来，而后利用数学方法计算每一层元素相对重要性次序的权值，最终通过所有层次间的总排序计算所有元素的相对权重并进行排序从而分析消费者决策。

FCE（Fuzzy Comprehensive Evaluation，模糊综合评判）20世纪80年代初，我国模糊数学领域的汪培庄教授提出了综合评判模型，并通过广大实际工作者的不断的补充发展，衍生出的适用于各种领域的评判方法。模糊综合评判的过程可简述为:决策者将价目标看成是由多重因素组成的因素集U，再设定这些因素所能选取的评审等级，组成评语的评判集合V,分别求出各单一因素对各个评审等级的模糊矩阵，然后根据各个因素在评价中的权重分配，通过模糊矩阵合成，求出评价的定量值。

但是这两种方法各有利弊：AHP中的能够准确的对决策定性，但其决策过程过程需要经过大量数据比对来最终通过概率确定权重；而FCE中虽然有很好的定量评价但是无法很好地对决策定性。请利用本案例的数据，尝试通过对二者的结合来实现对电商平台上热水器的购买决策分析。

AHP-FCE模型需要经历以下三个步骤，具体流程见图8。

1. 划分因素层。
2. 应用AHP构造消费者心理的隶属函数和因素权集合。
3. 对所求结果进行综合评判。

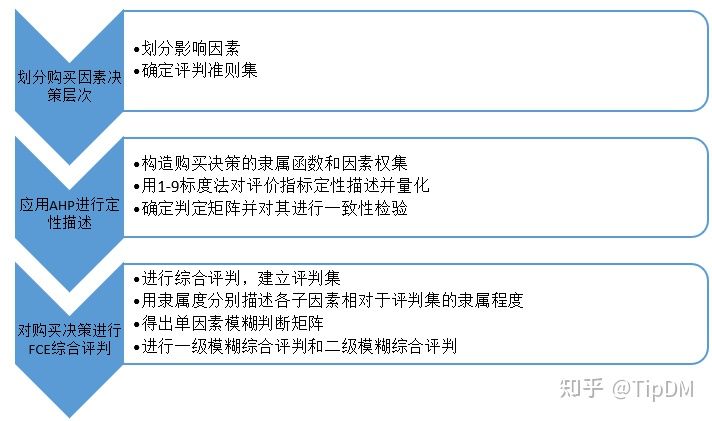


图8 AHP-FCE模型