电商平台评论数据情感分析

姓名	学号	专业
汤琼	10182100106	数据科学与大数据技术

概要

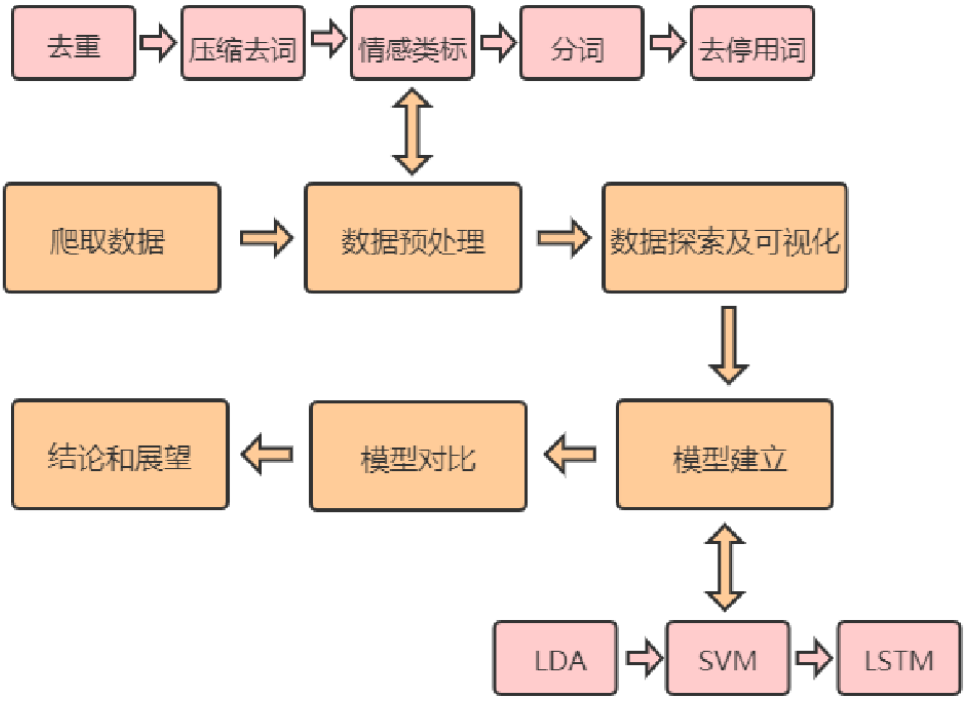
本文对电商平台上 *iPhone* 产品的用户评价文本数据进行预处理，分析和挖掘，分析用户的情感倾向，挖掘产品的优点和缺点，提炼产品的卖点，帮助生产者改进产品，帮助潜在用户挑选和归纳产品优缺点。本文首先介绍传统自然语言处理进行文本的预处理，然后分别采用**无监督的LDA主题模型**，**有监督的SVM机器学习模型**和**深度学习的LSTM模型**分别进行文本情感分析，实验表明**LSTM模型**的准确率

高达**94.95%**。

研究背景

随着互联网和电子商务的快速发展，网上购物对人们消费模式产生巨大影响。淘宝，天猫超市，京东等电商平台，涵盖了成千上万种类的产品，在没有接触实际产品的情况下选购商品，用户的评论信息具有很高的参考价值，潜在用户可以依据用户评价挑选合适的产品，店家和厂家也可以根据用户评论对产品和服务进行改进和提升。对成千上万条评论进行逐条浏览或者通过人工规则进行归纳都很费时费力，应用自然语言处理和文本挖掘的方法对产品评论进行自动挖掘，在很大程度上可以改善和提升用户体验。正是因为产品评论挖掘有着强大的现实应用意义和科学研究价值，评论情感分析`(sentiment analysis,SA)`也成为越来越多的研究者和工业界的兴趣焦点。

研究方法



- **机器学习**的常用方法，主要分为有**监督学习**和**无监督学习**。监督学习，就是人们常说的分类，通过已有的训练样本（即已知数据以及其对应的输出）去训练得到一个最优模型（这个模型属于某个函数的集合，最优则表示在某个评价准则下是最佳的），再利用这个模型将所有的输入映射为相应的输出，对输出进行简单的判断从而实现分类的目的，也就具有了对未知数据进行分类的能力。监督学习里典型的例子就是**KNN、SVM。无监督学习**则是另一种研究的比较多的学习方法，它与监督学习的不同之处，在于我们事先没有任何训练样本，而需要直接对数据进行建模。无监督学习里典型的例子就是**聚类**了。聚类的目的在于把相似的东西聚在一起，而我们并不关心这一类是什么。因此，一个聚类算法通常只需要知道如何计算相似度就可以开始工作了。
- 在有训练数据的情况下，**监督模型**总是比**无监督的预训练模型**表现的要好。其主要原因是监督模型对数据集的特性编码的更好。对数据了解地越透彻，学习的时间越少，模型建立也越准确。
- **LDA（隐狄利克雷分布）**是一种**无监督学习**的**主题概率生成模型**，输入是文档集合和主题个数，输出是以概率分布的形式呈现的主题，常用于主题建模、文本分类、观点挖掘等多个领域。
    - 它假定了一个前提：文档相当于一个词袋（bag-of-words），袋子中的词是独立可交换的，没有语法结构和顺序。
    - 其基本思想是：每个文档（Document）由多个主题（Topic）构成，每个主题都有对应的多个词（Word）来描述。

- 传统的神经网络结构比较简单：输入层-隐藏层-输出层；但只能单独地处理一个个的输入，前一个输入和后一个输入是完全没有关系的。但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的。而**循环神经网络** `RNN` (Recurrent Neural Network) 跟传统神经网络最大的区别在于每次会将前一次的输出结果，带到下一次的隐藏层中，一起训练。但我们容易发现，短期的记忆影响较大，长期的记忆影响就很小，这就是 RNN 存在的短期记忆问题。所以我们采用基于 RNN 的优化算法——`LSTM` **长短期记忆网络**。LSTM 长短期记忆网络采用一套灵活的逻辑——“只保留长序列数据中的**重要**信息，忽略不重要信息”。

实验创新点

1. 本文得原始数据均为作者爬取的电商网站上最新评价数据，具有一定的即时性和现实意义。
2. 区别于传统的结构化数据，本文处理分析非结构化数据，通过分词，去除停用词等方法，让计算机理解人类的语言。随着人工智能，计算机科学等学科发展，非结构语言将会成为人与计算机交互更重要的一部分。
3. 本文由浅入深地建立三种情感分析模型：**无监督的LDA主题模型**，**有监督的SVM机器学习模型**和**深度学习的LSTM模型**，分别进行文本情感分析并对三种模型进行对比。
4. 本文对数据集的划分采用随机分割和 **K-fold Cross Validation** K折交叉验证两种方法，使得结果不受测试集和训练集划分方法的影响。

数据集介绍

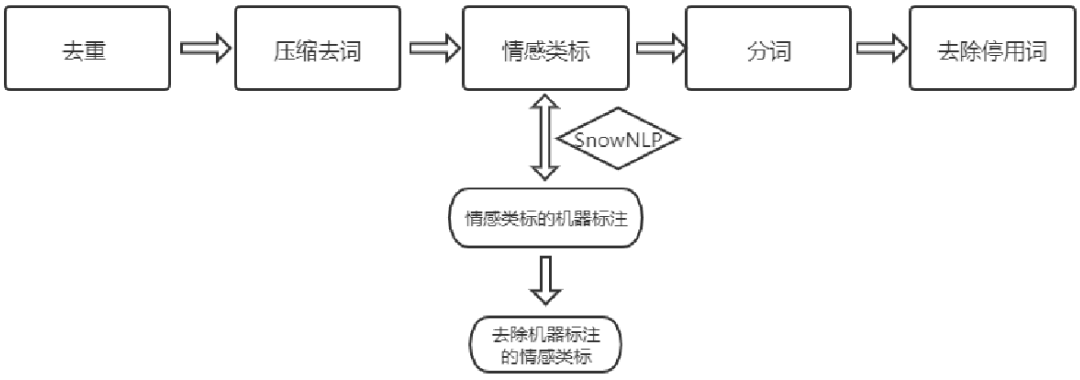
通过调用京东API接口采集到约一万条原始评论数据保存在 `iphone.csv` 文件中，数据集大小及评论数据样例内容如下。iphone 评论数据集包含11个字段，分别为：会员，级别，评价星级，评价内容，时间，点赞数，评论数，商品属性，页面网址，页面标题，采集时间。本文关注的是用户的评价内容，所以将数据集转成utf8编码txt格式导入，并将**“评价内容”**保存在 `iphone.txt` 文件中。

```
print(len(data))#数据集大小
```

```
9980
```

	会员	级别	评价星级	评价内容	时间	点赞数	评论数	商品属性	页面网址	页面标题	采集时间
0	1***p	NaN	star5	京东的快递很快！收到后急忙上手，外观特别漂亮、大气！屏幕音效很好！拍照技术更是无与伦比，效果...	43812.67778	10	17	暗夜绿色 256GB ...	https://item.jd.com/100004770237.html#none	【AppleiPhone 11 Pro Max】Apple iPhone 11 Pro Ma...	0.002025463
1	瓶子丶	PLUS 会员	star5	外形外观：这次的背面细磨砂质感非常特别，不容易留下指纹也不怕手汗，颜色主要的一大亮点就是暗夜...	43817.07986	9	0	暗夜绿色 256GB ...	https://item.jd.com/100004770237.html#none	【AppleiPhone 11 Pro Max】Apple iPhone 11 Pro Ma...	0.00202662

数据预处理



- 评论数据有 9980 条
- 文本评论中有大量价值很低甚至没有价值含量的评论，会对分析造成很大的影响，得到结果的质量也必然是存在问题的。

因此，需要在进行分析之前，进行文本的预处理，包括：

- 文本去重：例如：系统默认的好评；统一用户购买多个产品的相同/相似评论；垃圾评论
- 压缩长的重复词：例如：好好好好，一般一般
- 中文分词：`jieba`，结巴提供分词，词性标注，未登录词识别，支持用户词典等
- 清除停用词：在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些没有实际含义的字或词。

1）原始评论数据去重处理

数据去重的代码见附录1，去重结果如下，删除了573条重复评论，剩余9407条评论。

```
原始数据 9980 条评论，现在剩余 9407 条评论
删除了 573 条重复评论。
```

京东的快递很快！收到后急忙上手，外观特别漂亮、大气！屏幕音效很好！拍照技术更是无与伦比，效果杠杠的
外形外观：这次的背面细磨砂质感非常特别，不容易留下指纹也不怕手汗，颜色主要的一大亮点就是暗夜绿色，
自己的iPhone摔坏了，就买了这款暗夜绿的iPhone11 Pro Max，前一天晚上下单，第二天就送来了，很不错，
果然还是京东快，隔天收到！！ 绿色果然很骚气~而且很好看！！磨砂的手感~ 而且苹果系统也能从另外一台苹
手机已经使用一个月，说一下使用的感受，从iphone7plus开始一直用苹果，对我来说最好的就是不像用安卓机
关注很久了，领券下单的便宜几百块，手感很好，够重，送货速度很快，颜色也很新颖，今年的新配色，很喜欢
外形外观：不错。精致屏幕音效：效果完美。拍照效果：清晰运行速度：丝滑待机时间：很久。其他特色：系统
外形外观：真心没有感觉今年的iPhone三摄很丑 反而感觉很好看。去年的华为mate20不也是这个设计屏幕音效
外形外观：真的太完美，无可挑剔，摄像头就是最大的亮点，很好看，我喜欢这个磨砂的白色！屏幕音效：屏幕
外形外观：摄像头搭配紧凑，有气势。后背细腻柔顺，轻微磨纱感，强抗指纹，中框不锈钢质感，光鲜亮丽。屏
外形外观：没有想象的那么重，男孩子用正好，除了单手操作不太方便，银色太漂亮了，真香！屏幕音效：顶级
这次京东补贴活动很不错，手机也**了一个好价格，当然**后买的这款手机性能更强，拍照效果确实不错，视频
外形外观：金色，磨砂很好看屏幕音效：很赞，反应也很快拍照效果：拍照对于苹果来说就没得说了，但是新手

2） 原始评论数据压缩去词

用户的文本评论数据质量层次不齐，没有意义的文本数据很多。经过去重之后，还有很多评论需要处理，例如：

- “非常非常非常非常非常非常非常非常非常非常非常非常非常”
- “一般啦一般啦一般啦一般啦一般啦”
- “电池电量怎么这么这么这么这么少，呜呜呜呜”

压缩去词代码详见附录2，压缩去词的结果如下。

京东的快递很快！收到后急忙上手，外观特别漂亮、大气！屏幕音效很好！拍照技术更是无与伦比，效果杠杠的！
外形外观：这次的背面细磨砂质感非常特别，不容易留下指纹也不怕手汗，颜色主要的一大亮点就是暗夜绿色，配
自己的iPhone摔坏了，就买了这款暗夜绿的iPhone11 Pro Max，前一天晚上下单，第二天就送来了，很不错，应
果然还是京东快，隔天收到！！ 绿色果然很骚气~而且很好看！！磨砂的手感~ 而且苹果系统也能从另外一台苹果
手机已经使用一个月，说一下使用的感受，从iphone7plus开始一直用苹果，对我来说最好的就是不像用安卓机整
关注很久了，领券下单的便宜几百块，手感很好，够重，送货速度很快，颜色也很新颖，今年的新配色，很喜欢，
外形外观：不错。精致屏幕音效：效果完美。拍照效果：清晰运行速度：丝滑待机时间：很久。其他特色：系统依
外形外观：真心没有感觉今年的iPhone三摄很丑 反而感觉很好看。去年的华为mate20不也是这个设计屏幕音效：
外形外观：真的太完美，无可挑剔，摄像头就是最大的亮点，很好看，我喜欢这个磨砂的白色！屏幕音效：屏幕很
外形外观：摄像头搭配紧凑，有气势。后背细腻柔顺，轻微磨纱感，强抗指纹，中框不锈钢质感，光鲜亮丽。屏幕
外形外观：没有想象的那么重，男孩子用正好，除了单手操作不太方便，银色太漂亮了，真香！屏幕音效：顶级屏
这次京东补贴活动很不错，手机也**了一个好价格，当然**后买的这款手机性能更强，拍照效果确实不错，视频防
外形外观：金色，磨砂很好看屏幕音效：很赞，反应也很快拍照效果：拍照对于苹果来说就没得说了，但是新手机
这么稀缺的颜色居然让我抢到了，在搞活动的时候我抢到了，感觉很满意，颜色很漂亮，屏幕尺寸很合适，三个摄
外形外观：11 Pro max还是继承了前两代的一个总体外形设计，很漂亮，新加入了三摄镜头，很霸气，logo移到

3）情感类标的机器标注

可以使用 `SnowNLP` 对评论进行类标标注，代码如下，对文件进行情感得分处理。接着我们根据每条评论得分将评论分为：正面评论和负面评论，正向评论得分为正，负向评论得分为负，且分值的绝对值越大表明正向（负向）程度越大。我们再将 `txt` 文件内得分去除(去除机器标注的情感类标的代码见附件3），只保留评论内容。我们得到 505 条负面评论，8902 条正面评论 。

情感类标的机器标注

```
#情感类标的机器标注
import csv,codecs
from snownlp import SnowNLP

import time
start=time.time()

inputfile='data/iphone_process_2.txt'#评论文件
outputfile_pos='data/iphone_pos_1.txt'#评论处理后保存路径
outputfile_neg='data/iphone_neg_1.txt'#评论处理后保存路径

outf_pos=codecs.open(outputfile_pos,'wb',encoding='utf-8')
outf_neg=codecs.open(outputfile_neg,'wb',encoding='utf-8')

with codecs.open(inputfile,'r',encoding='utf-8') as inf:
    for line in inf:
        if not line.strip():#跳过空行
            continue
        sentscore=round((SnowNLP(line.strip()).sentiments - 0.5)*20)
        #print line,sentscore
        if sentscore>0.0:
            outf_pos.write(str(sentscore)+'\t'+line.strip()+'\n')
        elif sentscore<=0.0:
            outf_neg.write(str(sentscore)+'\t'+line.strip()+'\n')

outf_pos.close()
outf_neg.close()

end=time.time()
print('过程用时 %4.2f 秒' %(end-start))
```

```
负面评论有 505 条
正面评论有 8902 条
过程用时 0.12 秒
```

根据得分被归为正向评论且未去除机器标注的情感类标的txt文件

```
10	京东的快递很快！收到后急忙上手，外观特别漂亮、大气！屏幕音效很好！拍照技术更是无
10	外形外观：这次的背面细磨砂质感非常特别，不容易留下指纹也不怕手汗，颜色主要的一大
10	自己的iPhone摔坏了，就买了这款暗夜绿的iPhone11 Pro Max，前一天晚上下单，第二天
7	果然还是京东快，隔天收到！！ 绿色果然很骚气~而且很好看！！磨砂的手感~ 而且苹果系
10	手机已经使用一个月，说一下使用的感受，从iphone7plus开始一直用苹果，对我来说最好
10	关注很久了，领券下单的便宜几百块，手感很好，够重，送货速度很快，颜色也很新颖，今
10	外形外观：不错。精致屏幕音效：效果完美。拍照效果：清晰运行速度：丝滑待机时间：很
10	外形外观：真心没有感觉今年的iPhone三摄很丑 反而感觉很好看。去年的华为mate20不也
10	外形外观：真的太完美，无可挑剔，摄像头就是最大的亮点，很好看，我喜欢这个磨砂的质
10	外形外观：摄像头搭配紧凑，有气势。后背细腻柔顺，轻微磨纱感，强抗指纹，中框不锈钢
10	外形外观：没有想象的那么重，男孩子用正好，除了单手操作不太方便，银色太漂亮了，真
10	这次京东补贴活动很不错，手机也**了一个好价格，当然**后买的这款手机性能更强，拍照
```

去除机器标注的正向评论的txt文件

京东的快递很快！收到后急忙上手，外观特别漂亮、大气！屏幕音效很好！拍照技术更是无与
外形外观：这次的背面细磨砂质感非常特别，不容易留下指纹也不怕手汗，颜色主要的一大亮
自己的iPhone摔坏了，就买了这款暗夜绿的iPhone11 Pro Max，前一天晚上下单，第二天就
果然还是京东快，隔天收到！！ 绿色果然很骚气~而且很好看！！磨砂的手感~ 而且苹果系统
手机已经使用一个月，说一下使用的感受，从iphone7plus开始一直用苹果，对我来说最好的
关注很久了，领券下单的便宜几百块，手感很好，够重，送货速度很快，颜色也很新颖，今年
外形外观：不错。精致屏幕音效：效果完美。拍照效果：清晰运行速度：丝滑待机时间：很久。
外形外观：真心没有感觉今年的iPhone三摄很丑 反而感觉很好看。去年的华为mate20不也是
外形外观：真的太完美，无可挑剔，摄像头就是最大的亮点，很好看，我喜欢这个磨砂的白色
外形外观：摄像头搭配紧凑，有气势。后背细腻柔顺，轻微磨纱感，强抗指纹，中框不锈钢质
外形外观：没有想象的那么重，男孩子用正好，除了单手操作不太方便，银色太漂亮了，真香
这次京东补贴活动很不错，手机也**了一个好价格，当然**后买的这款手机性能更强，拍照效

4）分词

分词就是将句子、段落、文章这种长文本，分解为以字词为单位的数据结构，方便后续的处理分析工作。我们对正面情感和负面情感两个文本都进行中文分词，保存为两个文本文档，并与停用词一起作为主题模型（LDA）的输入。分词处理的代码见附件4。

- 可是为什么是分词？不是分字？

  因为词是表达完整含义的最小单位。字的粒度太小，无法表达完整含义，比如"鼠"可以是"老鼠"，也可以是"鼠标"。而句子的粒度太大，承载的信息量多，很难复用。比如"传统方法要分词，一个重要原因是传统方法对远距离依赖的建模能力较弱。"

下图即是分词后的结果，词与词之间是以空格为分隔符的。

京东 的 快递 很快 收到 后 急忙 上手 外观 特别 漂亮 大气 屏幕 音效 很 好 拍照 技术
外形 外观 这次 的 背面 细 磨砂 砂质 质感 非常 特别 不容 容易 留下 指纹 也 不怕 手 汗
自己 的 iPhone 摔坏 了 就 买 了 这款 暗 夜 绿 的 iPhone11 Pro Max 前一天 一天 晚上
果然 还是 京东 快 隔天 收到 绿色 果然 很 骚气 而且 很 好看 磨砂 的 手感 而且 苹
手机 已经 使用 一个 一个月 说 一下 使用 的 感受 从 iphone7plus 开始 一直 用 苹果 对
关注 很久 了 领 券 下单 的 便宜 几百 几百块 百块 手感 很 好 够重 送货 速度 很快 颜
外形 外观 不错 精致 屏幕 音效 效果 完美 拍照 效果 清晰 运行 行速 速度 丝 滑 待机
外形 外观 真心 没有 有感 感觉 今年 的 iPhone 三 摄 很丑 反而 感觉 很 好看 去年 的 华
外形 外观 真的 太 完美 无可 无可挑剔 挑剔 摄像 摄像头 像头 就是 最大 的 亮点 很 好
外形 外观 摄像 摄像头 像头 搭配 紧凑 有 气势 后背 细腻 柔顺 轻微 磨 纱 感 强 抗 指
外形 外观 没有 想象 的 那么 重 男孩 男孩子 孩子 用 正好 除了 单手 单手操作 操作 不
这次 京东 补贴 活动 很 不错 手机 也 了 一个 好 价格 当然 后 买 的 这款 手机 性能 更

5）清除停用词

在信息检索中，为节省存储空间和提高搜索效率，在处理自然语言数据（或文本）之前或之后会自动过滤掉某些没有实际含义的字或词，即停用词。

通常意义上，停用词(Stop Words)大致可分为如下两类：

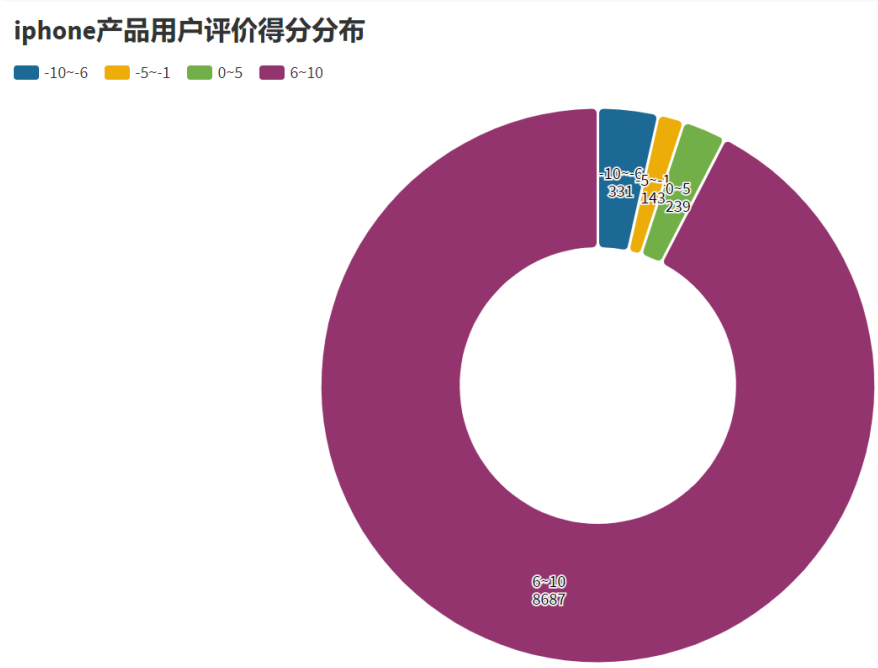
1、使用十分广泛，甚至是过于频繁的一些单词。比如英文的"i"、"is"、"a"、"the"，中文的"我"、"的"之类词几乎在每个文档上均会出现，查询这样的词搜索引擎就无法保证能够给出真正相关的搜索结果，难于缩小搜索范围提高搜索结果的准确性，同时还会降低搜索的效率。因此，在真正的工作中，Google和百度等搜索引擎会忽略掉特定的常用词，在搜索的时候，如果我们使用了太多的停用词，也同样有可能无法得到非常精确的结果，甚至是可能大量毫不相关的搜索结果。

2、文本中出现频率很高，但实际意义又不大的词。这一类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。如常见的"的"、"在"、"和"、"接着"之类。

	words	label	words_punc	words_cuts	words_cuts_stop	length	space_words
0	买的第二台了，第一台是暗夜绿色。外观方面，三个摄像头外观上吸睛度非常高。iPhone11 m...	0	买的第二台了，第一台是暗夜绿色。外观方面，三个摄像头外观上吸睛度非常高。iPhone11 m...	[买, 的, 第二台, 了, ，, 第一台, 是, 暗夜, 绿色, 。, 外观, 方面, ，, ...	[买, 第二台, 第一台, 暗夜, 绿色, 外观, 三个, 摄像头, 外观, 吸睛, 度, ...	62	买 第二台 第一台 暗夜 绿色 外观 三个 摄像头 外观 吸睛 度 高 iPhone11 m...
1	13号20点抢的，后来想改地址还不能改，不得不让快递小哥帮我改签，晚收到了一天。手机很喜欢，...	0	13号20点抢的，后来想改地址还不能改，不得不让快递小哥帮我改签，晚收到了一天。手机很喜欢，...	[13, 号, 20, 点, 抢, 的, ，, 后来, 想改, 地址, 还, 不能, 改, ...	[13, 号, 20, 点, 抢, 想改, 地址, 改, 快递, 小哥, 帮, 改签, 晚,...	32	13 号 20 点 抢 想改 地址 改 快递 小哥 帮 改签 晚 收到 手机 喜欢 绿色 磨...

数据探索及可视化

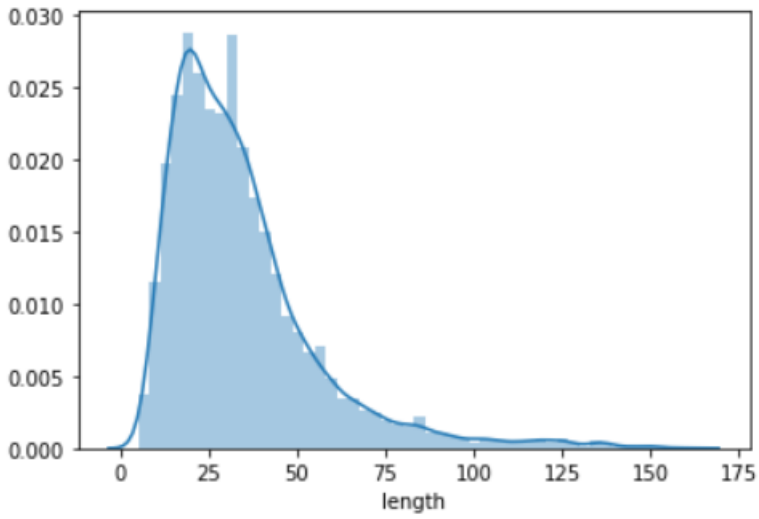
1） 用户评价得分分布情况



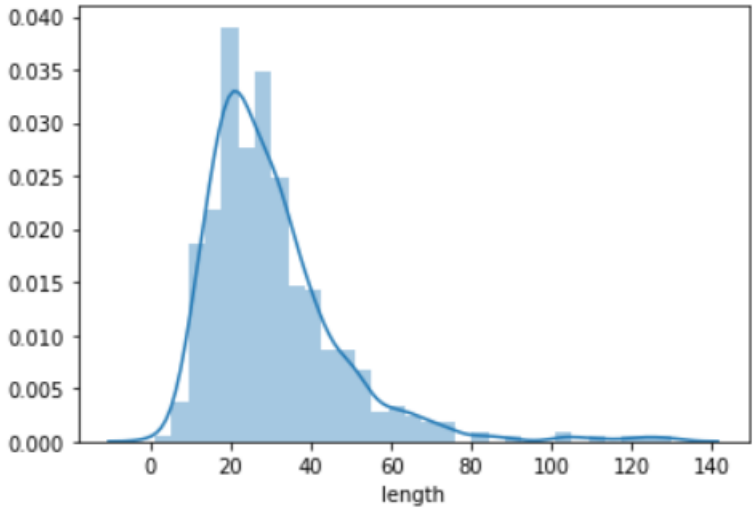
我们根据类标标注的评论得分绘制饼状图，由图可知，用户评论得分集中在6~10分，超过90%均为高分段正向评论，说明大部分客户对产品满意度较高。得分为-5~-1和0~5的评论数最少，说明对产品持较为中立态度的客户较少。

2）绘制正向，负向语言词数分布情况

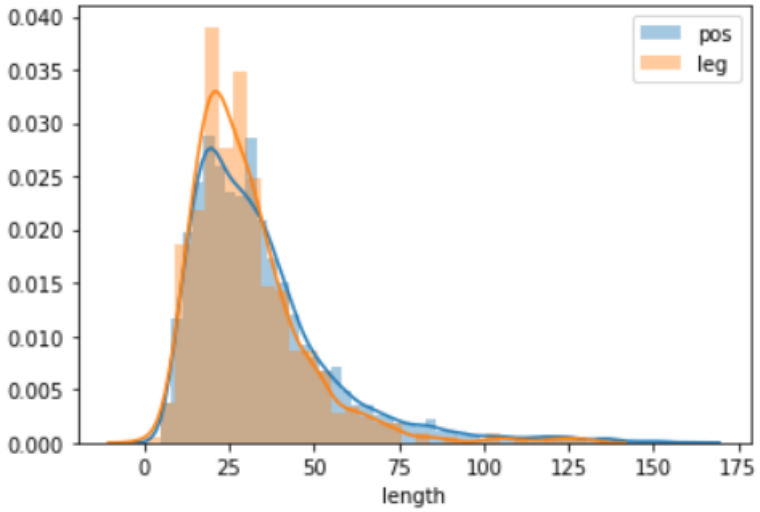
正向评论词数分布情况如下图，我们用一条曲线拟合柱状图，可观察到随着词数从零增长，词的分布（评论数）先急速增加后急速下降，到达一定程度时再缓慢下降，并在词数为25左右时达到峰值，说明正向语言词数集中在25个词的长度。



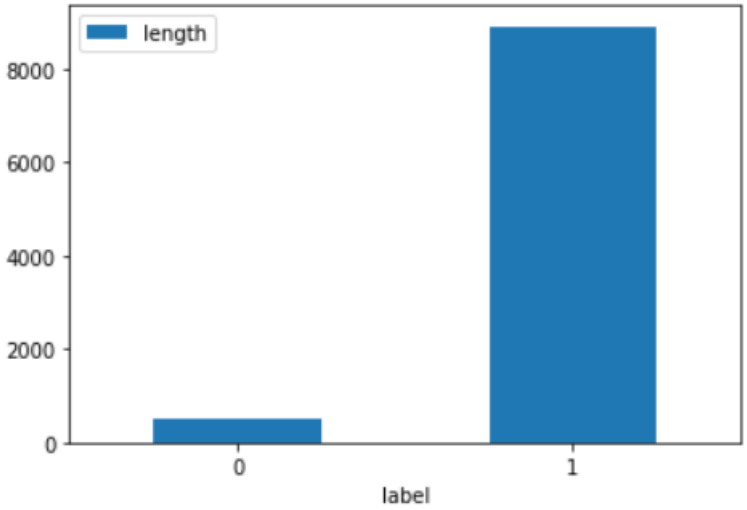
负向评论词数分布情况如下图，我们用一条曲线拟合柱状图，可观察到随着词数从零增长，词的分布（评论数）先急速增加后急速下降，到达一定程度时再缓慢下降，并在词数为25左右时达到峰值，说明负向语言词数集中在25个词的长度。



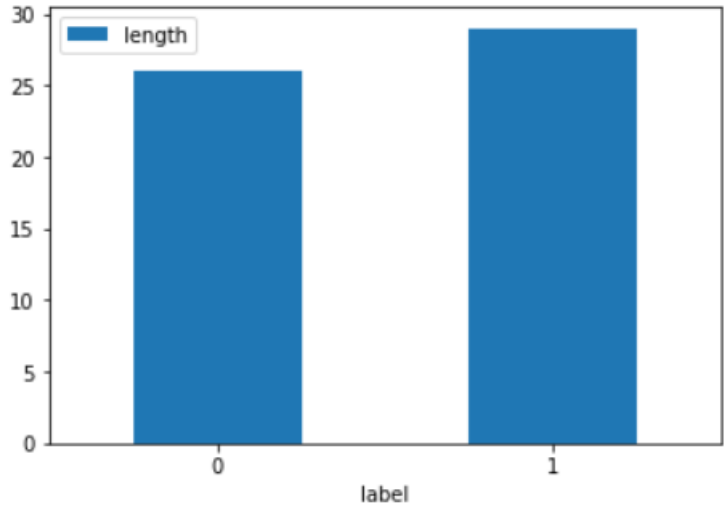
正向，负向语言词数分布情况如下图，我们将两张图叠加，可观察到，正负向评论分布总体增减趋势相近，且评论词数基本分布在 0~75 范围内。另外正负向评论分布均在词数为25左右时达到峰值，且此时负向评论多于正向评论条数。



3）正负向语料比例情况



由柱状图可知，正向评论数远多于负向评论数，正向评论数约9000条而负向评论数约500条，说明大部分用户对产品评价较高。



由柱状图可知情感倾向正负向语料长度中位数分布情况，看起来评价正面的话比较多。

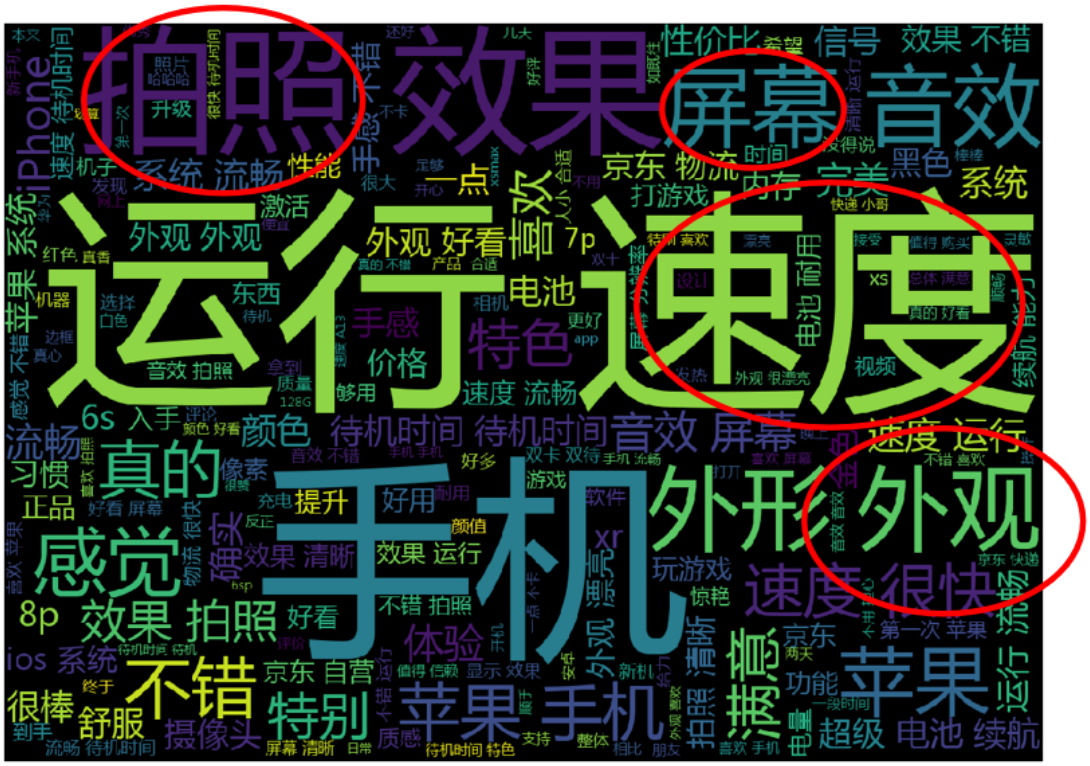
- 这里为什么要看中位数?

  在建立LSTM模型的时候，我们需要统一句子的向量长度，取中位数的长度为佳。长度超过30的字符串截断，长度不足30的字符串补零。

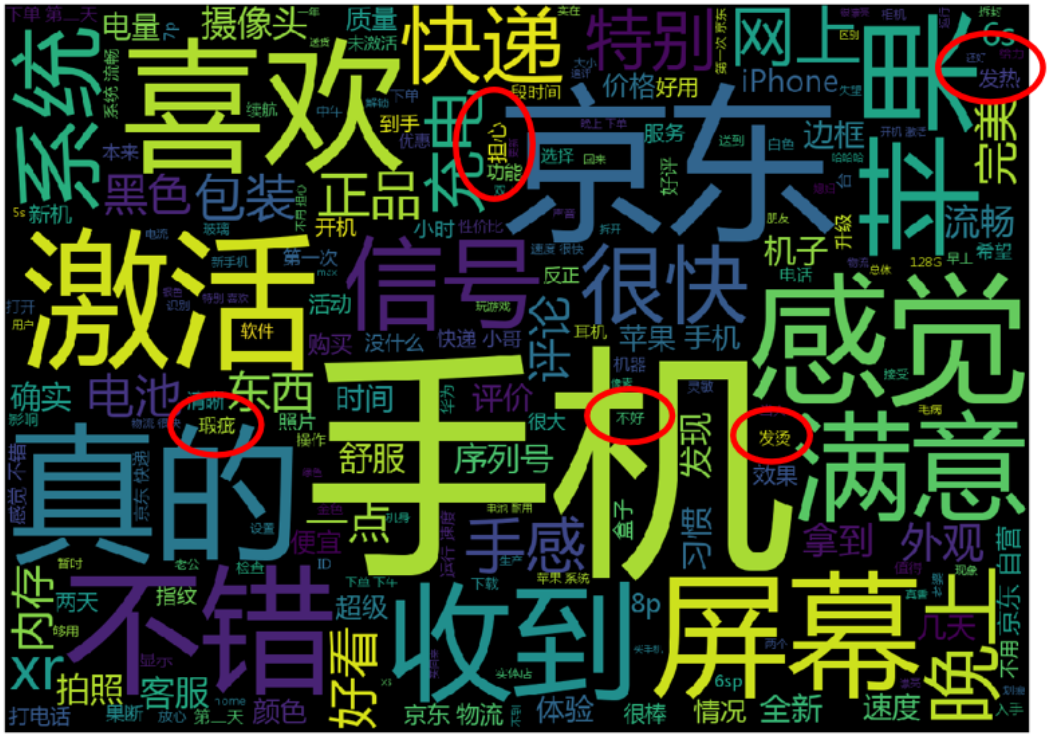
4）制作词云`wordcloud`

利用 python 的 wordcloud 库生成可视化图片，字的大小越大说明出现的次数越多，非常简约直观，生成效果如下：

正向语料词云，从词云中我们容易分析出 iPhone 的优势在于运行速度快，拍照效果好，屏幕清晰，外观好看等方面。

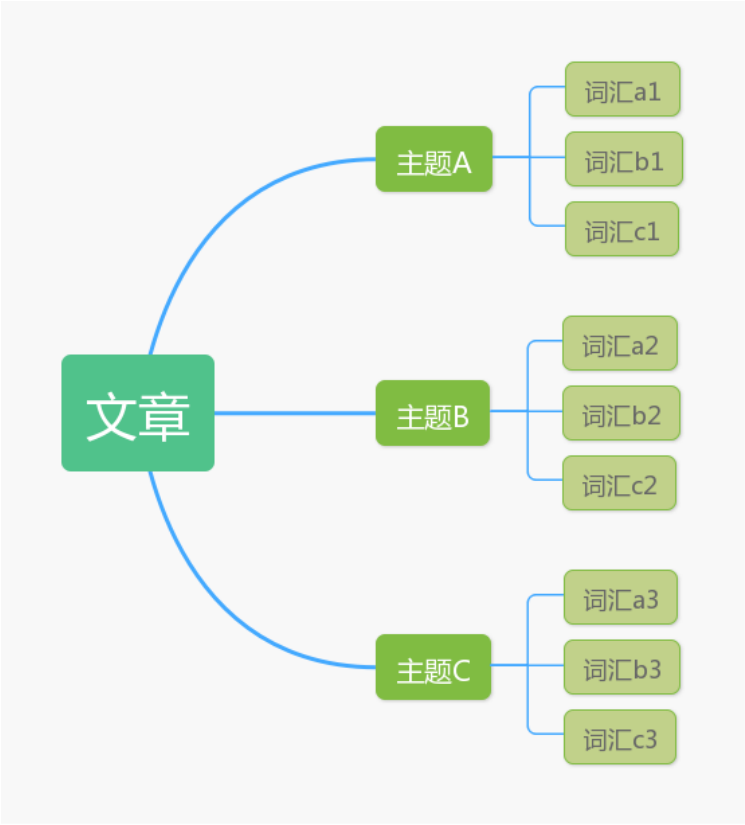


负向语料词云，从词云中我们容易分析 iPhone 存在的问题主要集中在手机质量，电量，外观瑕疵，发热发烫等方面。



建模建立

a）基于主题模型的情感分析



- LDA 是一种主题模型（Topic Model），LDA 认为同一个词，在不同的主题背景下，它出现的概率不同。文章和主题之间并不一定是一一对应的关系，也就是说文章可以有多个主题，一个主题可以在多篇文章中。同一个主题在不同文章中，他出现的比例（概率）是不同的。每个主题会对应一个词汇分布，而每个文档会对应一个主题分布。
- 因此，LDA定义两个分布：**主题与词汇分布**，**文章与主题分布**，即每个主题会对应一个词汇分布，每个文档会对应一个主题分布。主题K通过词汇的概率分布来反映。主题和词汇的分布就是多项式分布，而文章和主题之间的分布也是符合多项式分布。假设语料库中有词库V={v1,v2,...,vn},现在有K个主题，有M篇文章。在主题模型中，主题表现为一系列相关的单词的条件概率：
  - K1：{v1:0.01,v3:0.2,v4:0.3,v6:0.1,...}
  - K2：{v1:0.1,v2:0.05,v3:0.2,v5:0.3,...}
  - K3：{ v1:0.3,v2:0.1,v3:0.3,v7:0.1,...}

  而每个文档也是一个系列主题的概率分布：
  - M1：{K1:0.1,K2:0.5,K3:0.4}
  - M2：{K1:0.4,K2:0.3,K3:0.3}

```
#LDA 主题模型
start=time.time()

#参数初始化
negfile='data/iphone_neg_cut_1.txt'
posfile='data/iphone_pos_cut_1.txt'
stoplist='data/stoplist.txt'

#读入数据
neg=pd.read_csv(negfile,encoding='utf-8',header=None,engine='python')
pos=pd.read_csv(posfile,encoding='utf-8',header=None,engine='python')
stop=pd.read_csv(stoplist,encoding='utf-
8',header=None,engine='python',sep='tipdm')
#sep设置分割词，由于csv默认以半角逗号为分割词，而该词恰好在停用词表中，因此会导致读取出错
#所以解决办法是手动设置一个不存在的分隔符，如：tipdm
stop=[' ',' ']+list(stop[0])#pandas自动过滤了空格符，这里手动添加
```

```
neg[1]=neg[0].apply(lambda s:s.split(' ')) #定义一个分割函数，然后用apply广播
neg[2]=neg[1].apply(lambda x:[i for i in x if i not in stop])#逐词判断是否是停用词，思路同上

pos[1]=pos[0].apply(lambda s:s.split(' ')) #定义一个分割函数，然后用apply广播
pos[2]=pos[1].apply(lambda x:[i for i in x if i not in stop])

from gensim import corpora,models

#负面主题分析
neg_dict=corpora.Dictionary(neg[2])#建立负面词典
neg_corpus=[neg_dict.doc2bow(i) for i in neg[2]]#建立语料库
neg_lda=models.LdaModel(neg_corpus,num_topics=3,id2word=neg_dict)#LDA模型训练

pos_dict=corpora.Dictionary(pos[2])#建立正面词典
pos_corpus=[pos_dict.doc2bow(i) for i in pos[2]]#建立语料库
pos_lda=models.LdaModel(pos_corpus,num_topics=3,id2word=pos_dict)#LDA模型训练

end=time.time()
print('过程用时 %4.2f 秒'%(end-start))
```

经过 LDA 主题分析之后，评论文本被聚为3个主题，每个主题下面生成若干最有可能出现的词语以及相应频率。

下表展示了正面评价文本中的潜在主题

主题1	主题2	主题3
效果	手机	手机
运行	618	苹果
拍照	苹果	买
速度	兔	不错
屏幕	年	喜欢
时间	妈	速度
待机	键	京东
外观	128g	流畅
机时	32G	屏幕

- 主题1的高频特征词反映：iPhone 拍照效果好，待机时间长，外观好看。
- 主题2的高频特征词，关注点是：兔年，妈妈，128G，32G，反映 iPhone 可以作为新年礼物，iPhone 的内存空间也经常被关注。
- 主题3的高频特征词，关注点是：流畅，速度，屏幕，反映 iPhone 使用流畅，速度快等。

下表展示了负面评价文本中的潜在主题

主题1	主题2	主题3
手机	手机	手机
买	买	买
京东	京东	京东
激活	激活	喜欢
喜欢	说	感觉
感觉	苹果	激活
真的	不错	说
说	感觉	苹果
快递	屏幕	真的

- 高频特征词除了买，喜欢，不错，感觉，真的等情感词，提到了激活，快递等，主要反映：激活流程复杂，快递速度慢等问题。

分析

综合 iPhone 评价的主题和特征词我们可以看出：iPhone 用户的情感偏向绝大多数是正向的，其优势在于： 使用流畅，速度快，拍照效果好，外观好看等。劣势在于：激活流程复杂，发热等问题。

因此，我们给 iPhone 的建议是：在保证价格实惠的基础上，对电池发热进行改进，提高手机质量。

我们给京东的建议是：保证出售产品质量，避免瑕疵，翻新机等问题，为客户提供最优质的服务。

b）基于有监督的机器学习的情感分析

对于情感极性的判断，将目标情感分为三类：正，中，负。对训练文本进行标注，然后进行有监督的机器学习过程，并对测试数据用模型来预测结果。

抽取特征

这里使用TF-IDF特征

TF-IDF是一种用于信息检索与数据挖掘的常用加权技术，常用于挖掘文章中的关键词，TF-IDF有两层意思，一层是"词频"（Term Frequency，缩写为TF），另一层是"逆文档频率"（Inverse Document Frequency，缩写为IDF）。当有TF(词频)和IDF(逆文档频率)后，将这两个词相乘，就能得到一个词的TF-IDF的值。某个词在文章中的TF-IDF越大，那么一般而言这个词在这篇文章的重要性会越高，所以通过计算文章中各个词的TF-IDF，由大到小排序，排在最前面的几个词，就是该文章的关键词。

这里对不同验证分类方法，采用不同的机器学习性能评估指标，包括：精确率(Precision), 召回率(Recall)，准确率(Accuracy)。

- k折-交叉验证分类结果，第一行代表**精确率**(Precision),第二行代表**召回率**(Recall)。

```
[0.93426916 0.78141392 0.72652455 0.72849755 0.78667665]
[0.5541747  0.55249026 0.56537435 0.60273112 0.62888809]
```

- 随机分割分类结果（ 训练集：测试集=8：2），**准确率**(Accuracy)结果是**94.10%。**

```
0.9410201912858661
```

我们注意到：`TF-IDF`的优点是简单快速，而且容易理解。缺点这种计算**无法体现位置信息，无法体现词在上下文的重要性**。我们可以使用 **word2vec 算法**进行改进。

c）基于深度学习的情感分析

利用 `gensim` 中 Word2vec 工具，完成词向量转换工作，将处理好的数据导入 LSTM 模型中迭代，观察LOSS 变化情况，并预测其情感倾向分类。

- 传统的神经网络结构比较简单：输入层-隐藏层-输出层；但只能单独地处理一个个的输入，前一个输入和后一个输入是完全没有关系的。但是，某些任务需要能够更好的处理序列的信息，即前面的输入和后面的输入是有关系的。而循环神经网络 `RNN` (Recurrent Neural Network) 跟传统神经网络最大的区别在于每次会将前一次的输出结果，带到下一次的隐藏层中，一起训练。但我们容易发现，短期的记忆影响较大，长期的记忆影响就很小，这就是 RNN 存在的短期记忆问题。所以我们采用基于 RNN 的优化算法—— `LSTM` 长短期记忆网络。
- LSTM 长短期记忆网络采用一套灵活的逻辑——“只保留长序列数据中的重要信息，忽略不重要信息”。
- 所有递归神经网络都具有神经网络的链式重复模块。在标准的 RNN 中，这个重复模块具有非常简单的结构，例如只有单个 tanh 层。LSTM 也具有这种类似的链式结构，但重复模块具有不同的结构。不是一个单独的神经网络层，而是四个，并且以非常特殊的方式进行交互。LSTM 的关键是细胞状态。LSTM 可以通过所谓“门”的精细结构向细胞状态添加或移除信息。门可以选择性地以让信息通过。它们由 S 形神经网络层和逐点乘法运算组成。S 形网络的输出值介于 0 和 1 之间，表示有多大比例的信息通过。0 值表示“没有信息通过”，1 值表示“所有信息通过”。一个 LSTM 有三种这样的门用来保持和控制细胞状态。

词向量 Word2vec

词向量也叫词嵌入，即 word vector 和 word embedding 。对每个词，采用分布式表达。

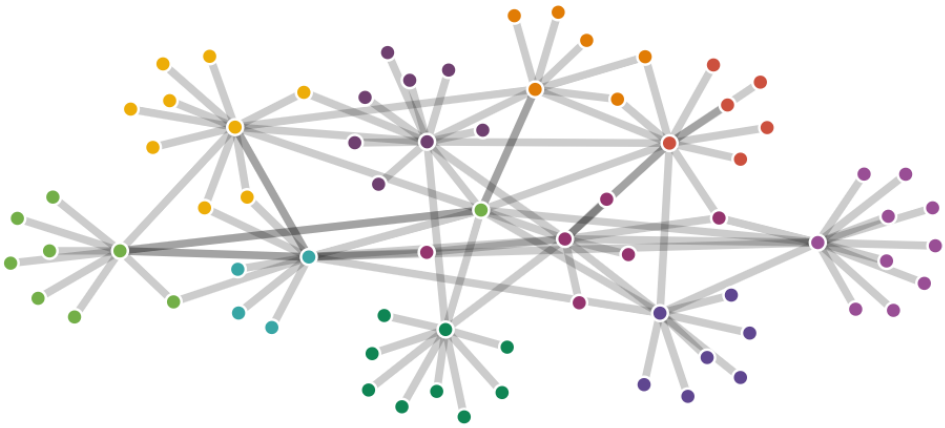
Word2Vec就是用**高维向量**表示词语，并把相近意思的词语放在相近的位置，而且用的是实数向量（不局限于整数）。我们只需要有大量的某语言的语料，就可以用它来训练模型，获得词向量。词向量除了可以用高维向量表示词语，解决了词语的**多方向发散问题**，另一方面词向量可以方便做聚类，用欧氏距离或余弦相似度都可以找出两个具有相近意思的词语。这就相当于解决了“**一义多词**”的问题。

我们取出“iphone”这个词的词向量，即分布式表达，举例说明词向量的表示形式。

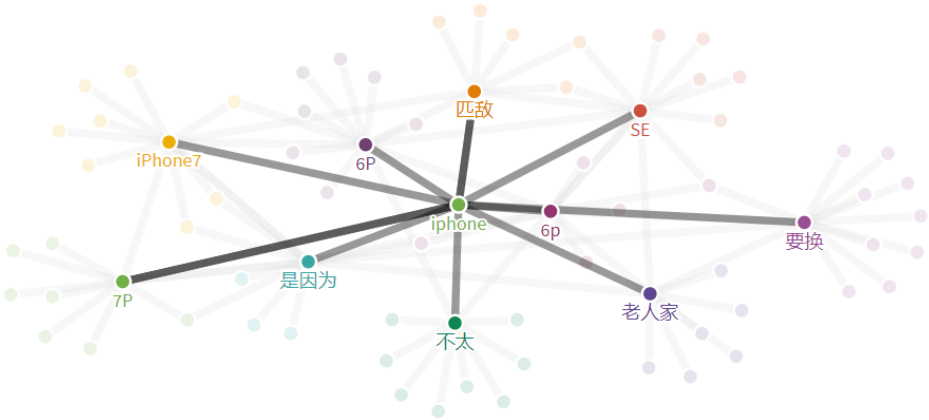
```
array([-0.0064337 ,  0.33697966, -0.29727182,  0.50665987,  0.09924648,
        0.21075751,  0.4500451 ,  0.21910872, -0.00657773, -0.29038838,
       -0.27390715,  0.3655081 ,  0.44505465, -0.2610968 , -0.23819071,
       -0.30867726, -0.22981317, -0.43251273,  0.31729874,  0.15589485,
        0.5149074 ,  0.5769746 , -0.5054964 , -0.41975382,  0.14659981,
        0.7246967 , -0.00952156, -0.22919741,  0.20214903,  0.16462949,
        0.25057828,  0.07186987, -0.07943951, -0.46323988, -0.14734201,
       -0.26472506, -0.23503052,  0.4148697 , -0.85247797, -0.1777881 ,
       -0.2768219 ,  0.23725773, -0.74311274,  0.08460457, -0.03036502,
       -0.04514771, -0.15989473,  0.07648604,  0.31774998, -0.6089955 ,
        0.17147924,  1.3448246 ,  0.6813474 ,  0.20060256, -0.02759781,
        0.4572234 , -0.42881775, -0.01273543,  0.18960841,  0.37905625],
      dtype=float32)
```

找到语义空间上与 `iphone` 最接近的十个词：`iPhone7`，`7P`，`6p`，`是因为`，`匹配`，`老人家`，`不太`，`6P`，`SE`，`要换`。并接着找分别与十个词在语义空间上最接近的十个词（具体代码和输出结果见附录），画出 network charts 网络图（如下图），展现语义空间上不同词之间的关系。

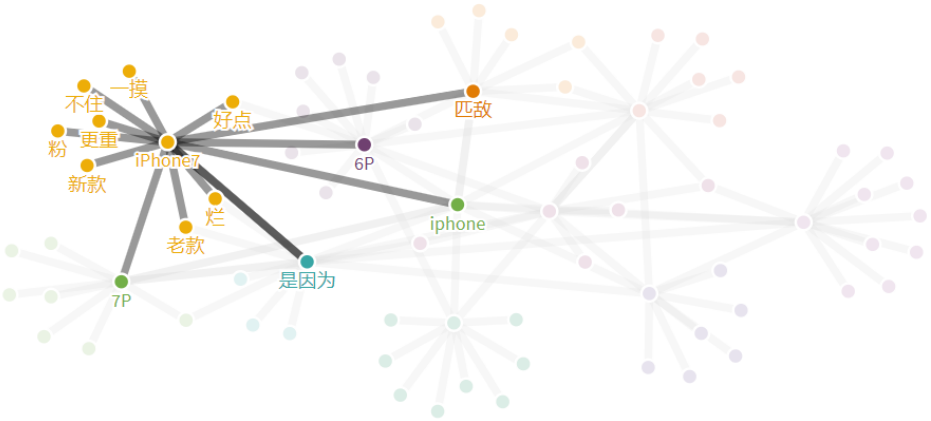
图中的点代表一个词语，相同颜色的点在语义空间上较为接近，图（b）展现的是与 iphone 语义最接近词语的 network charts，图（c）与 iPhone7 语义最接近词语的 network charts。



（a）语义空间上最接近词语的 network charts



（b）与 iphone 语义最接近词语的 network charts



（c）与 iPhone7 语义最接近词语的 network charts

从词向量到句子向量

接下来要解决的问题是：我们已经分好词，并且已经将词语转换为高维向量，那么句子就对应着词向量的集合，也就是矩阵，类似于图像处理，图像数字化后也对应一个像素矩阵；可是模型的输入一般只接受一维的特征，一个比较简单的想法是将矩阵展平，也就是将词向量一个接一个，组成一个更长的向量。但一个句子包含多个词，句子的长度不同，因此，直接把句子所有词的向量拼接，则句子的向量长度就不统一，后续算法无法进行。因此，需要把句子固定相同的长度，即固定字符串的长度，做法如下：

- 为了避免太多字符串为空，我们确定字符串长度为30（约为评论词数中位数）
- 长度超过30的字符串截断
- 长度不足30的字符串补零

将截取后的词按照训练好的w2v词向量转换，并以训练集大小：测试集大小=8：2，分割数据集。

建立LM神经网络模型

建立模型训练，耗时时间可能较长，如果需要快速查看效果，可以将 nb_epoch 参数修改的小一点。

- 神经网络的第一层需要接受一个关于输入数据 shape 的参数（30，60），这里表示句子长度为30个字，每个字的词向量为60维度。
- 后面各个层则可以自动推导出中间数据的 shape, 因此不需要为每个层都指定这个参数。

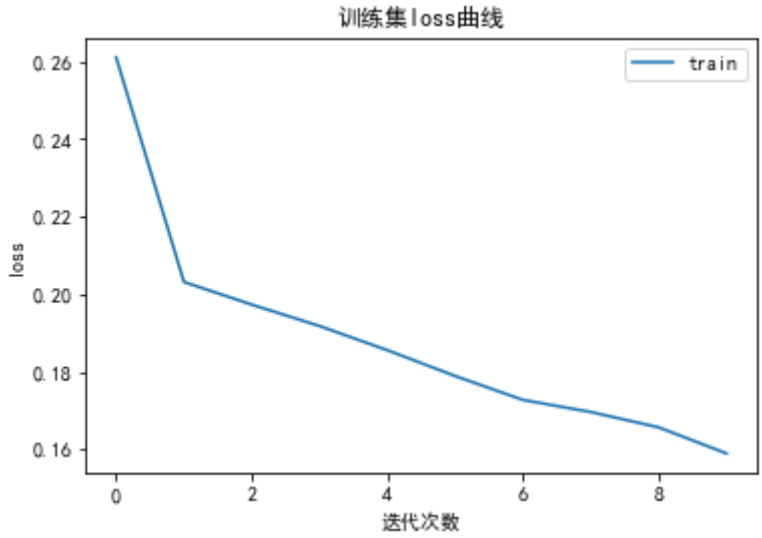
  神经网络模型建立的具体代码如下：

```
#构建LM神经网络模型
from keras.models import Sequential#导入神经网络初始化函数
from keras.layers.core import Dense,Activation#导入神经网络层函数，激活函数

import time
start=time.time()

model=Sequential()
model.add(LSTM(22,input_shape=(30,60),dropout=0.5,activation='sigmoid'))#22个神经元的LSTM，
#第一层需要接受一个关于输入数据shape的参数（30，60），后面的各层不需要指定
```

```
model.add(Dense(2,activation='softmax'))#输出层为2个节点，使用softmax激活函数
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=
['accuracy'])#编译模型，用adam方法求解

print(model.summary())
print('Fit Model..')

#神经网络训练
history=model.fit(x_train_modify,y_train_modify,batch_size=64,epochs=10)

end=time.time()
print('过程用时 %4.2f秒' % (end-start))
```

```
%matplotlib inline
import matplotlib as mpl
from matplotlib import pyplot

mpl.rcParams['font.sans-serif']=['SimHei']#['Microsoft YaHei']#指定为默认字体
mpl.rcParams['axes.unicode_minus']=False#解决保存图象是负号'-'显示为方块的问题

pyplot.plot(history.history['loss'],label='train')
plt.title(u'训练集loss曲线')
plt.xlabel(u'迭代次数')
plt.ylabel(u'loss')
plt.legend()#显示图例
plt.show()#显示作图结果
#pyplot.plot(history.history['acc'],label='acc')
```

损失函数（loss function）是用来估量模型的预测值与真实值的不一致程度，它是一个非负实值函数,损失函数越小，模型的鲁棒性就越好。我们观察得到的训练集 loss 曲线，可知随着迭代次数增加，loss 值越小，证明模型的准确度越高，鲁棒性越好。



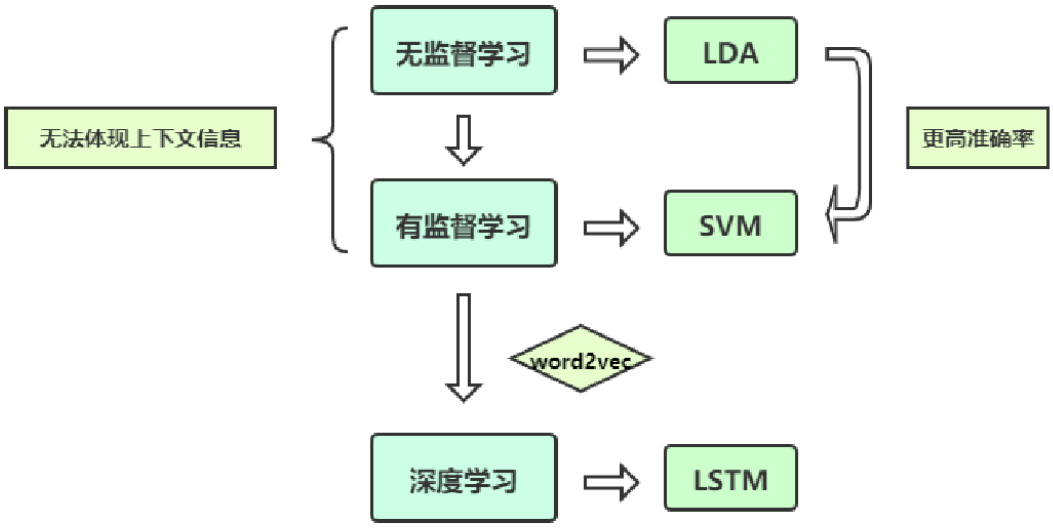
预测结果

准确率结果是：

```
0.9495217853347503
```

准确率结果约为 94.95%, 对比前两种的准确率，LSTM的准确率有所提高。

模型对比



机器学习的常用方法，主要分为**有监督学习**和**无监督学习。LDA（隐狄利克雷分布）**是一种**无监督学习**的**主题概率生成模型**，其基本思想是：每个文档（Document）由多个主题（Topic）构成，每个主题都有对应的多个词（Word）来描述。LDA主题模型可以较为直观的看出产品评论中正向和负向文本，但模型准确率不够高，进一步我们从文本中设计并抽取重要的特征，进行标注，进行**有监督的机器学习**过程，可获得更高准确率。

但无论是有监督的机器学习还是无监督的机器学习，都存在一个缺点，即无法体现词的位置信息，无法体现词在上下文的重要性。我们可以使用**word2vec 算法**进行改进，并引入**LSTM长短期记忆网络。**LSTM可以解决RNN存在的短期记忆问题，其采用一套灵活的逻辑——"只保留长序列数据中的**重要**信息，忽略不重要信息"。

经过三种模型的对比验证，我们得到采用**深度学习**进行文本挖掘，情感分析的准确度最高。

总结与展望

1. iPhone 产品的用户情感偏向大多为正向，其优势在于： 使用流畅，速度快，拍照效果好，外观好看等。劣势在于：激活流程复杂，发热等问题。因此我们给 iPhone 的建议是：在保证价格实惠的基础上，对电池发热等问题进行改进，提高手机质量。我们给京东的建议是：保证出售产品质量，避免瑕疵，翻新机等问题，为客户提供最优质的服务。
2. 我们所建立的三种情感分析模型：**无监督的LDA主题模型**，**有监督的SVM机器学习模型**和**深度学习的LSTM模型** ，无监督的 LDA 主题模型可以较为直观的看出 iPhone 产品的优势和劣势，在此基础上引入有监督的机器学习模型可以进一步提高模型准确性，然后这两种模型都无法解决词的位置信息问题，于是利用词向量模型进行改进，并引入LSTM长短期记忆网络模型，通过实验检验发现：**LSTM 长短期记忆网络模型准确率最高。**
3. 我们的实验还存在一些改进之处：
    - 爬取的**数据量**不够多，增大数据量可以使模型更加完善，准确率也会提高。
    - 对评论进行类标标注的时候，**人工标注**的准确率会高于机器标注，但会大大提高人工成本。
    - 如果想进一步提高准确率，**停用词表**需要进一步补充整理；建立 word2vec 的时候，语句长度，输出维度参数等可以进一步调参试试。
    - 本实验结果只能返回**二维情感极性**：正向和负向，无法提炼细节情绪。经过进一步了解，基于深度学习模型，以海量带标签文本数据为训练集，选取更细节的特征对象，再加上逻辑加权、词库校验等方式训练出AI引擎可以至多识别出中文文本中多达18种细节情绪，每种所识别的情绪还会附加情感浓度值，以反映这些情绪的程度属性。

4. 文本情感分析是非常重要的一类文本挖掘任务，是**人工智能**的重要研究方向，具有很高的**学术价值。**同时可以应用到消费决策，舆情分析，个性化推荐，观点立场分析，公共安全，电商产品评论；包含中文和英文的产品评论数据，例如电商数据，微博评论，豆瓣电影评论，推特评论，酒店评论，餐馆评论，金融股票情感预测，机器人情绪对话等，具有很高的**商业价值。**另外，在**自然语言处理**领域，除文本外，**语音情感识别与合成**、**人脸表情识别**等方面也会成为人工智能理解人类语言情感的重要技术。

附录

1. 去重处理

```
#去重处理
inputfile='data/iphone.txt'#评论文件
outputfile='data/iphone_process_1.txt'#评论处理后保存路径
data=pd.read_csv(inputfile,encoding='utf-8',header=None)
l1=len(data)
data=pd.DataFrame(data[0].unique())
l2=len(data)
data.to_csv(outputfile,index=False,header=False,encoding='utf-8')
print('原始数据 %s 条评论，现在剩余 %s 条评论' % (l1,l2))
print('删除了 %s 条重复评论。' %(l1-l2))
```

2. 压缩去词

```
#数据压缩去词
#-*- coding: utf-8 -*-
import codecs
inputfile = 'data/iphone_process_1.txt' #评论文件
outputfile = 'data/iphone_process_2.txt' #评论处理后保存路径
f = codecs.open(inputfile ,'r','utf-8')
f1=codecs.open(outputfile,'w','utf-8')
fileList = f.readlines()
f.close()
for A_string in fileList:
    temp1= A_string.strip('\n')        #去掉每行最后的换行符'\n'
    temp2 = temp1.lstrip('\ufeff')
    temp3= temp2.strip('\r')
    char_list=list( temp3)
    list1=['']
    list2=['']
    del1=[]
    flag=['']
    i=0
    while(i<len(char_list)):
        if (char_list[i]==list1[0]):
            if (list2==['']):
                list2[0]=char_list[i]
            else:
                if (list1==list2):
                    t=len(list1)
                    m=0
                    while(m<t):
                        del1.append( i-m-1)
                        m=m+1
                    list2=['']
                    list2[0]=char_list[i]
```

```
                else:
                    list1=['']
                    list2=['']
                    flag=['']
                    list1[0]=char_list[i]
                    flag[0]=i
        else:
            if (list1==list2)and(list1!=[''])and(list2!=['']):
                if len(list1)>=2:
                    t=len(list1)
                    m=0
                    while(m<t):
                        del1.append( i-m-1)
                        m=m+1
                    list1=['']
                    list2=['']
                    list1[0]=char_list[i]
                    flag[0]=i
            else:
                if(list2==['']):
                    if(list1==['']):
                        list1[0]=char_list[i]
                        flag[0]=i
                    else:
                       list1.append(char_list[i])
                       flag.append(i)
                else:
                    list2.append(char_list[i])
        i=i+1
        if(i==len(char_list)):
           if(list1==list2):
                    t=len(list1)
                    m=0
                    while(m<t):
                        del1.append( i-m-1)
                        m=m+1
                    m=0
                    while(m<t):
                        del1.append(flag[m])
                        m=m+1
    a=sorted(del1)
    t=len(a)-1
    while (t>=0):
        #print(char_list[a[t]])
        del char_list[a[t]]
        t=t-1
    str1 = "".join(char_list)
    str2=str1.strip() #删除两边空格
    f1.writelines(str2+'\r\n')
f1.close()
```

3. 去除机器标注的情感类标

```
#去除机器标注的情感类标
start=time.time()

#参数初始化
```

```
inputfile1='data/iphone_neg_1.txt'
inputfile2='data/iphone_pos_1.txt'
outputfile1='data/iphone_neg_2.txt'
outputfile2='data/iphone_pos_2.txt'

data1=pd.read_csv(inputfile1,encoding='utf-8',header=None,delimiter="\t")#读入数据
data2=pd.read_csv(inputfile2,encoding='utf-8',header=None,delimiter="\t")

data1=pd.DataFrame(data1[1].astype(str).replace('.*?\d+?\\t ', ''))#用正则表达式修改数据
data2=pd.DataFrame(data2[1].astype(str).replace('.*?\d+?\\t ', ''))

data1.to_csv(outputfile1,index=False,header=False,encoding='utf-8')#保存结果
data2.to_csv(outputfile2,index=False,header=False,encoding='utf-8')

l1=len(data1)
l2=len(data2)
print('负面评论有 %s 条' % (l1))
print('正面评论有 %s 条' % (l2))

end=time.time()
print('过程用时 %4.2f 秒'%(end-start))
```

4. 分词

```
start=time.time()

inputfile1='data/iphone_pos_2.txt'
inputfile2='data/iphone_neg_2.txt'

outputfile1='data/iphone_pos_cut_1.txt'
outputfile2='data/iphone_neg_cut_1.txt'

data1=pd.read_csv(inputfile1,encoding='utf-8',header=None)
data2=pd.read_csv(inputfile2,encoding='utf-8',header=None)

myseg=lambda s: ' '.join(jieba.cut(s,cut_all=True))#自定义简单分词函数

data1=data1[0].apply(myseg) #通过广播形式分词，加快速度
data2=data2[0].apply(myseg) #通过广播形式分词，加快速度

data1.to_csv(outputfile1,index=False,header=False,encoding='utf-8')#保存结果
data2.to_csv(outputfile2,index=False,header=False,encoding='utf-8')#保存结果

end=time.time()
print('过程用时 %4.2f 秒' %(end-start))
```

5. 语义空间上`iphone`最接近的十个词及分别与十个词在语义空间上最接近的十个词

```
emotion_w2v.most_similar(positive=['iphone'], topn=10)
```

```
[('iPhone7', 0.9788615107536316),
 ('7P', 0.9762703776359558),
 ('6p', 0.975593626499176),
 ('是因为', 0.9720801115036011),
 ('匹敌', 0.9715782999992371),
 ('老人家', 0.9674206972122192),
 ('不太', 0.9671282768249512),
 ('6P', 0.9614686965942383),
 ('SE', 0.9611095190048218),
 ('要换', 0.9599297046661377)]
```

```
emotion_w2v.most_similar(positive=['iPhone7'], topn=10)
```

```
[('是因为', 0.9886237382888794),
 ('更重', 0.9850231409072876),
 ('新款', 0.9819875955581665),
 ('6P', 0.9819157123565674),
 ('粉', 0.981374979019165),
 ('老款', 0.9809890985488892),
 ('不住', 0.9809398651123047),
 ('一摸', 0.9808741807937622),
 ('烂', 0.9805139303207397),
 ('好点', 0.9800957441329956)]
```

```
emotion_w2v.most_similar(positive=['7P'], topn=10)
```

```
[('是因为', 0.9823002815246582),
 ('8P', 0.9787402153015137),
 ('plus', 0.9787334203720093),
 ('去年', 0.9767342805862427),
 ('iphone', 0.976270318031311),
 ('之间', 0.9707934260368347),
 ('MacBook', 0.97022545337677),
 ('iPhone7', 0.9699951410293579),
 ('系列', 0.9669215679168701),
 ('8plus', 0.9666322469711304)]
```

```
emotion_w2v.most_similar(positive=['6p'], topn=10)
```

```
[('iPhone6s', 0.9840444922447205),
 ('不太', 0.9804149866104126),
 ('6P', 0.9801666736602783),
 ('替换', 0.9798564910888672),
 ('SE', 0.9782952070236206),
 ('要换', 0.978243887424469),
 ('是因为', 0.9779538512229919),
 ('办法', 0.9777188301086426),
 ('盯', 0.9770842790603638),
 ('太小', 0.9762251377105713)]
```

```
emotion_w2v.most_similar(positive=['是因为'], topn=10)
```

```
[('iPhone7', 0.9886235594749451),
 ('盯', 0.9861828684806824),
 ('11PRO', 0.9833865761756897),
 ('7P', 0.9823001623153687),
 ('要换', 0.9811389446258545),
 ('置换', 0.9810377359390259),
 ('老款', 0.9809580445289612),
 ('Xs', 0.9802694320678711),
 ('烂', 0.9790747165679932),
 ('8plus', 0.9790146350860596)]
```

```
emotion_w2v.most_similar(positive=['匹敌'], topn=10)
```

```
[('SE', 0.9726783037185669),
 ('iphone', 0.9715781807899475),
 ('完爆', 0.9702816605567932),
 ('6P', 0.9679367542266846),
 ('多一些', 0.9662744998931885),
 ('IPhone', 0.9649409651756287),
 ('换到', 0.9644379615783691),
 ('iPhone7', 0.9643067717552185),
 ('没太大', 0.9637556076049805),
 ('相', 0.9632092118263245)]
```

```
emotion_w2v.most_similar(positive=['老人家'], topn=10)
```

```
[('最新款', 0.9833931922912598),
 ('难度', 0.9809530973434448),
 ('要换', 0.9762029647827148),
 ('6p', 0.9749271273612976),
 ('互动', 0.9743231534957886),
 ('搞定', 0.9739415645599365),
 ('是因为', 0.973861813545227),
 ('没法', 0.9734196662902832),
 ('SE', 0.9734094738960266),
 ('三款', 0.9732581377029419)]
```

```
emotion_w2v.most_similar(positive=['6P'], topn=10)
```

```
[('se', 0.9916541576385498),
 ('iphonex', 0.989536464214325),
 ('划', 0.9877583980560303),
 ('甚', 0.9865277409553528),
 ('强迫症', 0.9864469170570374),
 ('原先', 0.985785186290741),
 ('有点儿', 0.9855287075042725),
 ('好点', 0.9855245351791382),
 ('住', 0.9854432344436646),
 ('携带', 0.9854058027267456)]
```

```
emotion_w2v.most_similar(positive=['SE'], topn=10)
```

```
[('多一些', 0.9872984886169434),
 ('换成', 0.9837996959686279),
 ('换到', 0.9833689332008362),
 ('受不了', 0.9818791151046753),
 ('已有', 0.9812853336334229),
 ('6p', 0.9782951474189758),
 ('6P', 0.9781906604766846),
 ('11promax', 0.9775813221931458),
 ('迹象', 0.977526843547821),
 ('iPhone6s', 0.9775264859199524)]
```

```
emotion_w2v.most_similar(positive=['要换'], topn=10)
```

```
[('不买', 0.9866960048675537),
 ('第一台', 0.9861973524093628),
 ('iphone8plus', 0.9861468076705933),
 ('真香机', 0.9855000972747803),
 ('咬咬牙', 0.9853658080101013),
 ('当初', 0.9849907755851746),
 ('iPhone6s', 0.9847708940505981),
 ('身边', 0.9843974113464355),
 ('这部', 0.984228253364563),
 ('推出', 0.9836162328720093)]
```