# 基于文本挖掘的评论数据分析

## 背景

从文本数据中发现有效信息，为产品的更新迭代提供依据。

## 目的

发现商品评论中对商品的一些负面情感，挖掘商品的一些不足，建议等。

## 项目准备

进行文本挖掘的首要任务是分词，因此我第一步就是从jieba这个python库入手，了解jieba库的功能。

Jieba的分词模式有许多，例如全模式、精确模式、搜索引擎模式，我用上的只有精确模式。Jieba的add\_word()和suggest\_freq(segment, tune=True)函数从描述上来看说可以在程序中动态修改词典，使词能或不能被分出来，但是我在使用后发现，它的操作是一次性的，也就是说当你退出重新登录后以前添加的词就没了，这也是后来自己建立词库的原因，还有一点jieba.Tokenizer(dictionary=DEFAULT\_DICT)是加载自定义分词器，与词库无关，jieba自己有默认的分词器，因此不要改动。

分完词后要做的是去除停用词，去除停用词的目的是为了提高一些无意义词对后续分析的影响，也为了提高效率。起初停用词词库我是从网上下载的大家公认的比较好的词库，但是我在使用过程中发现，这些词库主要是适用于长文本，因为去除的停用词中会包含一些对于短文本来说还是比较重要的信息，例如否定词不，不是等等，还有一些有情感倾向的成语，例如物美价廉，导致起初的关键词提取或者情感判断的结果都比较糟糕，后来就自己建立了停用词词库。

jieba的git地址：<https://github.com/fxsjy/jieba>

## 基于业务需求的评论分析

需求1：对评论按照关键词分类，提取关键词

需求2：找出评论中的好评与差评

需求3：找出价值比较高的评论

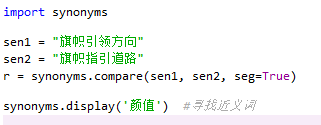
### 评论分类

关于评论分类，最开始我想到的是用聚类的方法对评论分类。文本的聚类与数字的聚类不同，目前的聚类方法都是针对数值型的数据，因此要实现文本的聚类首先要将词语向量化，我是用word2vec（原理讲解：https://blog.csdn.net/wangyangzhizhou/article/details/77073023）实现词语的向量化，对向量化后的数据用KNN（kNN算法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别）算法聚类。这种方法虽然可以将评论分类，但是属于被动的聚类，无法决定分类后的结果。

后来考虑采用计算词语之间的相似度来寻找出描述同一关键词的评论，例如当我想要寻找描述质量的评论时，通过寻找质量的同义词来找出相似的评论。开始时使用的是synonyms这个库，它可以计算两个句子的相似度，也可以用来寻找词语的同义词。

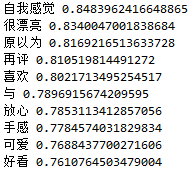
经过测试发现，只有当两个句子的结构相似，并且词语在字典中有记录时才较为准确，当句子较长，并且句法复杂时，它的效果也很差；再说它寻找同义词的功能，只有当它的字典中存在该词时，才能寻找出近义词，例如“质量”的近义词为：

使用代码如下：

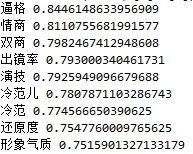


从结果中可以看出，它计算出的近义词对于评论数据挖掘来说毫无用处，因为我更希望它能找出像“品质”，“质地”这样的词（但即使是能找出这样的词，也还达不到我们想要的目标），并且当词语不存在它的词典中时，就无法计算，例如“颜值”这个词。最糟糕的是我们无法对它的词库进行修改，因此这个库被直接放弃。

后来了解到用word2vec也可以计算词语的相似度，并且它的模型可以训练，便开始尝试。首先我用自己店铺的5w条评论数据去训练词向量模型，然后去寻找关于“颜值”的近义词，得到的结果如下：



然后我又从网上下了一个基于百度百科的数据训练的模型进行测试，结果为：



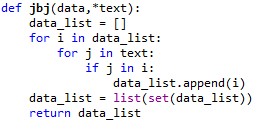
这并不是我想要的结果，后来我去查word2vec的原理，发现这个近义词是计算词向量余弦距离最接近的几个词，可以理解为根据模型的理解计算出与“颜值”最近的一些词。然后我计算了一下“颜值”和“外观”，它们的相似度为：



可以看出它们的相似度水平较低，因此不能根据这个来寻找词的近义词。而网上的关于寻找近义词的方法就是关于这些，因此只能自己想办法找出我想要的那一类评论。

最后使用的方法就是建立一个关键词词库，通过在所有评论中查找关键词的方法来对评论进行分类。

代码如下：



### 关键词提取

关于关键词提取，网上有许多的关于关键词提取的方法，例如TF-IDF，TEXTRANK，LDA，以及LDA的一些进阶算法等等。起初看到这么多方法时很高兴，又有东西搬过来用了，但是经过测试发现，例如一句评论为“可别可爱！虽然孩子还小，估计也就能用碗，杯子和沙拉杯还得等上了一岁才能用，都给孩子留着”得到的结果：

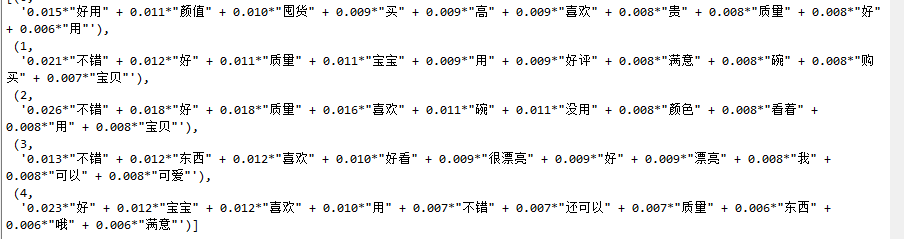


可以看出得到的关键词没有可读性，并且也没把我们认为关键的词列出来，例如“可爱”。

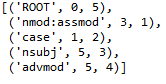
研究了原理后发现，TF-IDF算法的名字叫"词频"（TF）和"逆文档频率"（IDF），由名字也能知道，这个算法是根据词频，和逆文档频率计算得出的，词频的意思就是词出现的频率，一篇文章中出现的越多的词是关键词的可能性越大，而逆文档的意思是如果这个词在其他文章中出现的次数越多，它的词频就会下降。也就是说在这个文档中出现的多，在其他的文档中出现的少的词会被作为关键词，而评论数据是一个相对独立的短文本数据，并不要考虑其他评论的词是什么，因此转而研究TEXTRANK算法。还是那句评论，测试的结果为：



我发现很多评论用该算法根本无法提取关键词，在之后又了解到LDA算法（主题模型算法），得到的结果也与前面的类似，区别在于LDA算法得到的主题有比例，例如：



它的结果依然不是自己想要的，之后在想如何提取关键词时，想到了如果能提取句子的主干也相当于提取了句子的关键词。句子主干提取是利用stanfordcorenlp的句法依存关系来实现，目前并没有有效的提取句子主干的算法，该库也就是帮助分析了句子的依存关系，例如一句评论是“保温碗的质量真的非常好”，它的句法依存关系为：



其中root表示中心词，nsubj表示名词主语等等（还有相关的一些词汇http://www.hankcs.com/nlp/chinese-sentences-svo-java-extraction.html#appendix），我们要做的只能是通过这些关系来制定逻辑规则已达到提取句子主干的目的，我尝试写了一些规则，也只能处理一些简单句，例如“背带的质量真的非常好”的处理结果为：



代码地址如下（https://github.com/s19293949/-/blob/master/%E6%8F%90%E5%8F%96%E4%B8%BB%E5%B9%B2.py）

如果要实现所有句子主干的提取会非常困难，要对句法结构有非常深入的了解，后来注意到淘宝的标签就是一个关键词提取的方法。

在后续的学习中了解到淘宝标签制作可以用两种方法实现：

1. 用关联规则推荐的算法去制作标签：它的思路是先找出一些出现的比较多的名词，然后用关联规则算法去寻找出现那些名词后出现的比较多的形容词，该方法的缺陷是1.不能计算词频，2.输出的一些关键词有些可能并不是我们想要的，3.并不能实现对关键词的分类。
2. 用语义相似度匹配的方法制作标签：它的思路是先对语料分词，找出语料中所有的名词与形容词，然后遍历所有的名词和形容词，用word2vec算法计算每个名词与每个形容词之间的相似度，根据相似度排序得到关键词。该方法有一定的逻辑性，但是也有一些随机的成分，并且也没有解决词频以及按关键词分类的问题。

因此最后自己想了一个方法，就是通过将评论分句，然后寻找出包含某关键词的部分，再去匹配关键词之后的字段，找出修饰词来达到目的。

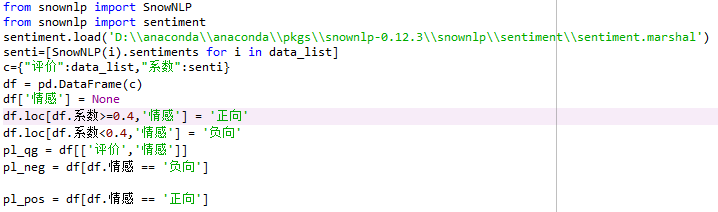
代码如下：



## 评论情感判断

**算法一**：关于句子的情感判断，网上流传的比较多的方法就是通过snownlp库来进行情感判断，该库的使用方法也很简单，直接调用sentiment模块就能对文本进行判断。起初我用它自带的模型进行情感判断，效果不理想。后来我用自己店铺的数据作为样本集训练了一个模型，终于使预测的准确率达到了一个客观的水平。

代码如下：



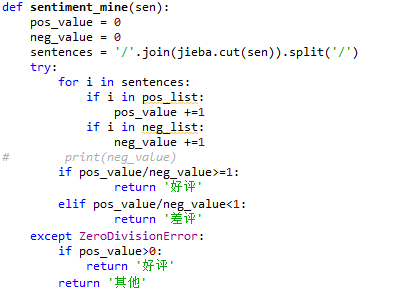
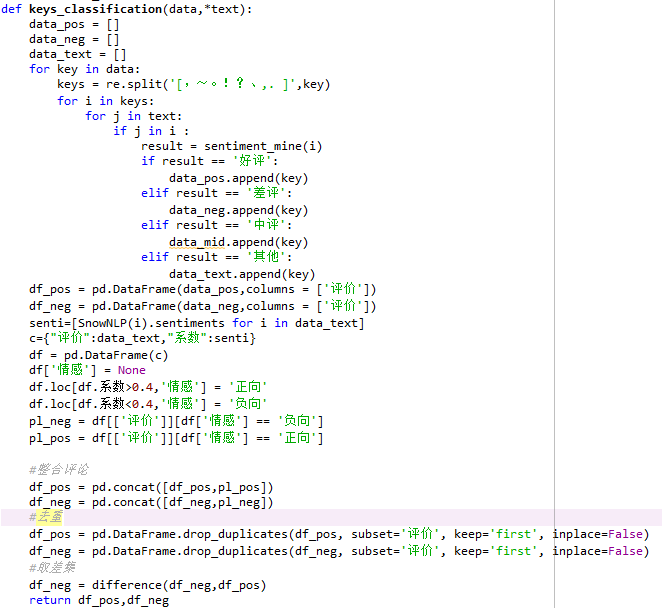
之后又碰到了一个问题，虽然目前来说对总体评论的情感判断的准确率能达到90%以上，但是一旦对评论分类后，针对某类评论的准确率就会比较低，例如“碗还行吧，只是内胆根本取不下来，按照客服说的方法也没取下来，就这样将就用吧。物流慢到极点！！！” 这条评论就描述了三个方面，一个是产品，一个是客服，一个是物流，并且每一部分的情感也不一样，因此如果直接进行判断很容易出错。我的解决方法是制定一个新的情感判断规则：

**算法二**：将评论分句，然后针对关键词对情感词进行判断，继而判断该评论针对这一类别的情感性质。

但是这种处理方法也有一个问题，那就是有些评论它的关键词和情感词根本不在一个部分里，比如说“用了两天，跟想象的一样。天气开始凉了，有了这个就再也不用热饭了。还有客服&rdquo;怀特&rdquo;态度也特别好，喜欢的可以放心买啦”它对客服的描述是分开的，因此针对这一情景，我制定的规则是将我的两部分情感判断算法结合在一起：1.当评论中包含有明确的情感词时，用算法二判断评论情感；当分句中的关键词无情感词时，则采用算法一模型来进行情感判断。

Snownlp库的git如下：https://github.com/isnowfy/snownlp

代码如下：



除此之外为了提升模型判断的准确率，还研究了基于循环神经网络的lstm算法（git如下https://github.com/joliph/Text-Classify）该算法是基于RNN循环神经网络的基础上被提出来的，其优点在于对序列数据建模，并且当数据前后有较强的关联性时，有很好的效果。该算法的原理以及代码实现都很困难，目前仍在学习中。