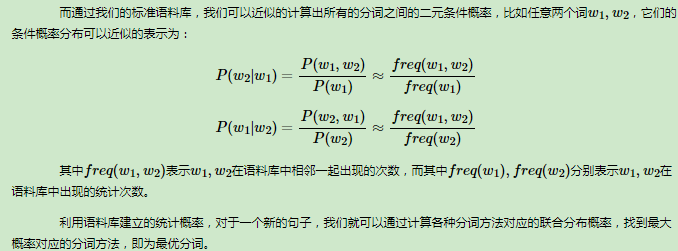
文本挖掘预处理的第一步，关键一步“分词”



分词常用算法维特比算法：

具体参考：http://www.cnblogs.com/pinard/p/6677078.html

而在做了分词后，如果我们是做文本分类聚类，则后面关键的特征预处理步骤有向量化或向量化的特例Hash Trick

随后需要进行预处理,常用的为tfidf

1. 所谓概率函数就是要在整个样本空间分配概率值，概率值总和为1
2. 一个完备的概率空间应该由样本空间，概率函数和事件域这三部分组成，在统计自然语言处理中，我们的目标就是为建立的模型定义一个符合上述条件的概率空间。
3. 随机变量有一个取值的范围，因此我们避免了直接和事件本身打交道，而是处理代表它们的随机变量的值，为了更清楚的描述随机变量的特性，我们引入概率密度函数的概念，它表示随机变量取不同值的概率。

4.期望值是一个随机变量的各个取值的平均值。一个随机变量的方差描述该随机变量的值偏离其期望值的程度。在[概率论](https://baike.baidu.com/item/%E6%A6%82%E7%8E%87%E8%AE%BA" \t "https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6%E6%9C%9F%E6%9C%9B/_blank)和统计学中，数学期望(mean)（或[均值](https://baike.baidu.com/item/%E5%9D%87%E5%80%BC" \t "https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6%E6%9C%9F%E6%9C%9B/_blank)，亦简称期望）是试验中每次可能结果的[概率](https://baike.baidu.com/item/%E6%A6%82%E7%8E%87" \t "https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6%E6%9C%9F%E6%9C%9B/_blank)乘以其结果的总和，是最基本的数学特征之一。它反映随机变量平均取值的大小。需要注意的是，期望值并不一定等同于常识中的“期望”——“期望值”也许与每一个结果都不相等。期望值是该变量输出值的[平均数](https://baike.baidu.com/item/%E5%B9%B3%E5%9D%87%E6%95%B0" \t "https://baike.baidu.com/item/%E6%95%B0%E5%AD%A6%E6%9C%9F%E6%9C%9B/_blank)。期望值并不一定包含于变量的输出值集合里。

假设一个质地均匀的骰子，即每一面出现的概率都为1/6，则它的期望值为21/6，这个期望值表示多次掷骰子实验得到的总点数和除以实验的次数。（相当于均值）



1. 我们用联合概率分布来描述样本空间中多个随机变量的分布，设两个离散型随机变量X和Y，他们的联合密度函数可写为

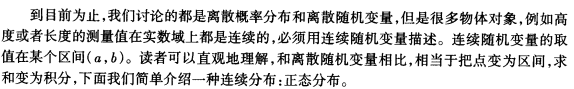
条件概率密度公式



1. 离散分布函数：二项分布

当重复一个只有两种输出（假设0和1）的实验，并且各次实验之间相互独立时，我们就说实验的结果服从二项分布。二项分布在统计自然语言处理中应用的很广泛，例如估计英文文本中含有单词the的句子所占的百分比。

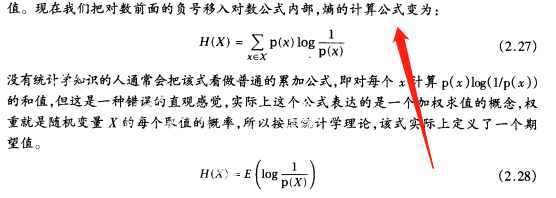
1. 连续分布函数：正态分布



1. 极大似然估计

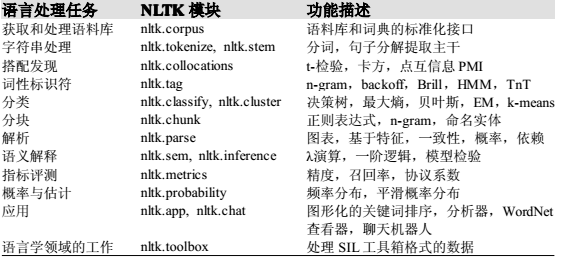
仍以抛硬币为例，假设我们重复10次，其中8次结果是正面朝上的，从一个频度论者的角度来看，他会认为10次实验中8次正面朝上是一种极大似然估计。

1. 熵表示单个随机变量的不确定性的均值，随机变量的熵越大，它的不确定性越大，也就是说能正确估计其值的概率越小。熵值度量了随机变量所包含的信息量的大小。熵可以用于衡量事件不确定的大小，事件所透漏出来的信息越多，其熵值会越小。

10.词干提取不注意词法会丢失一些信息

11.应该用什么特征来指导设计标注集呢？通常，一个标注集同时依据分类目标特征和预测特征来进行编码，分类目标特征告诉用户关于一个词的语法类别的有用信息，预测特征是对预测上下文中其它词语特性有用的特征，他们通常是相互交叉的，但是没有必要等同看待。

12.NLTK各模块功能介绍



13.NLTK 频率分布类中定义的函数

               例子                                                        描述  
   fdist = FreqDist(samples)                     创建包含给定样本的频率分布  
   fdist.inc(sample)                                                 增加样本  
   fdist['monstrous']                                     计数给定样本出现的次数  
   fdist.freq('monstrous')                                     给定样本的频率  
   fdist.N()                                                               样本总数  
   fdist.keys()                                             以频率递减顺序排序的样本链表  
   for sample in fdist:                                    以频率递减的顺序遍历样本  
   fdist.max()                                                         数值最大的样本  
   fdist.tabulate()                                                 绘制频率分布表  
   fdist.plot()                                                        绘制频率分布图

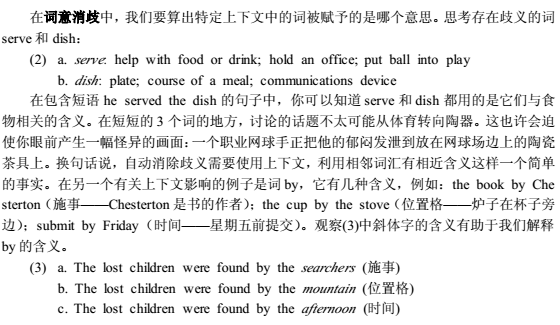
   fdist.plot(2,cumulative=True) 绘制累积频率分布图，前面的数值是选择几个词（选填的不填是所有）

注意返回的fdist是一个字典，键是文本中的词，值是在文本中该词出现的个数

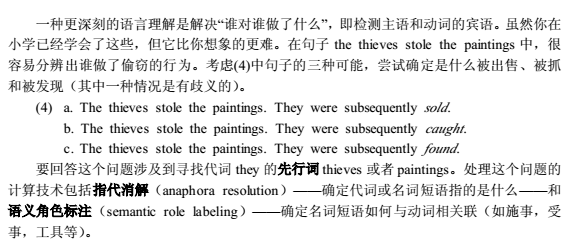
*#频率统计，统计文本中每个词出现的次数*all\_words =FreqDist(w.lower() **for** w **in** word\_tokenize(**"I'm foolish foolish man"**))  
sort\_dict=sorted(all\_words.items(),key=operator.itemgetter(1),reverse=**True**)  
**for** item **in** sort\_dict:  
 print(item[0],**'-->'**,item[1])

具体参考：<https://www.cnblogs.com/liangqihui/p/6730991.html>

14.词意消歧



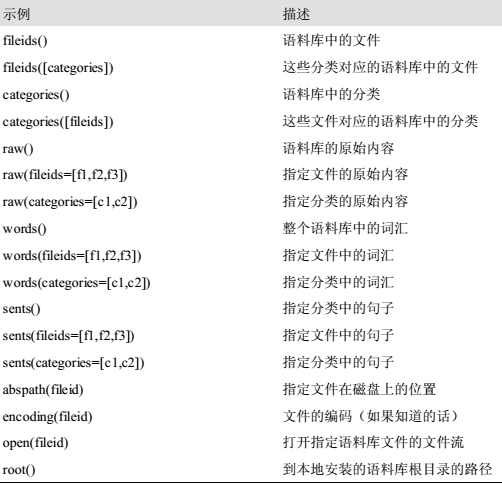
15.指代消解



16.NLTK语料库用法介绍

<http://www.cnblogs.com/itdyb/p/5900195.html>

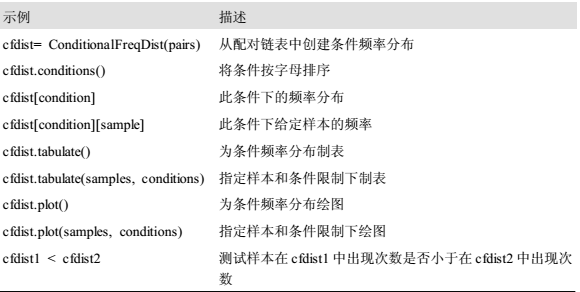
17.NLTK中基本的语料库函数



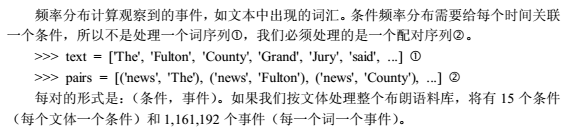
18.条件频率分布ConditionalFreqDist

条件频率分布是频率分布的集合，每个频率分布有不同的‘条件’，这个条件通常是文本的类别。

NLTK中条件频率分布所用函数



19.条件和事件



20.把字符串转换为NLTK的特殊Text格式后可进行一些操作

text=nltk.Text(text)把text转换为NLTK独有的Text

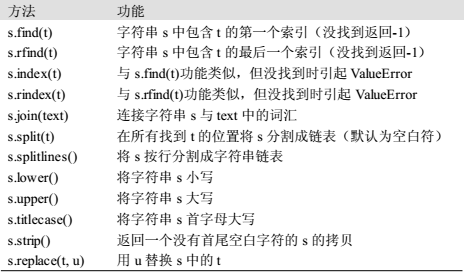
text.concordance("monstrous")

concordance:搜索text中的monstrous

text.similar("monstrous")

查找text中与monstrous相关的所有词语

21.操作字符串的一些方法



22.正则表达式

在python中使用正则表达式需要导入import re

re.search(p,s)检查字符串s中是否有模式p

正则表达式中的$用于匹配单词的末尾

例如[w for w in wordlist if re.search(‘ed$’,w)]代表从wordlist列表中取出以ed结尾的元素组成新的列表

.是通配符，匹配任何单个字符。

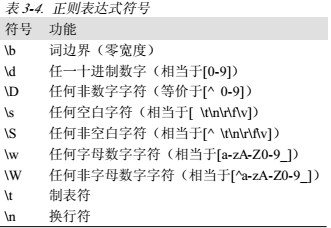
^匹配字符串的开始。

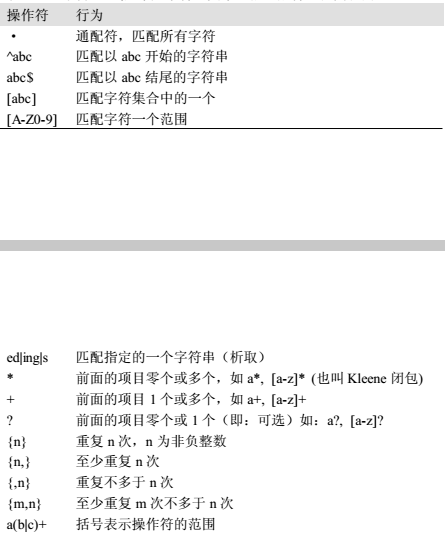
？表示前面的字符是可选的e-?mail即可匹配email也可匹配e-mail

+匹配的是前面项目的一个或多个实例。例如m+可匹配m,mm,m...

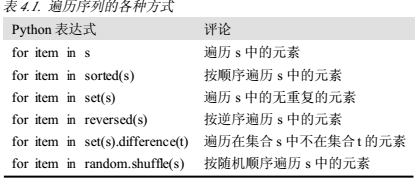
\*匹配的是前面项目的0个或多个实例。

正则表达式前面加r的作用是，告诉python解释器按照字面表示对待字符串，而不去处理正则表达中包含的反斜杠字符。





23.对字符串，列表，元组这些序列的操作



24.word2vec应用场景

<https://www.zhihu.com/question/25269336>

API案例：

<https://blog.csdn.net/a1368783069/article/details/52025764>

官方API：

<https://radimrehurek.com/gensim/models/deprecated/word2vec.html>

25.jieba中常用函数

<https://blog.csdn.net/qq_27231343/article/details/51898940>

26.情感分析与文本主题分析

情感分析就是对例如评论的文本进行分类，把其分为好评和坏评等。

首先应该把文本中的词转换为词特征向量作为文本的特征进行分类聚类等处理。可以把文本中所有不重复的词组成一个词袋，然后统计单词的词频

**from** sklearn.feature\_extraction.text **import** CountVectorizer

vectorizer=CountVectorizer()

*vectorizer.fit\_transform(corpus)*

如上是进行词频统计

有些常用单词出现的评率很高，但是其重要性并不是很高，故应该对词向量进行TF-IDF处理。

IDF:IDF反应了一个词在所有文本中出现的频率，如果一个词在很多的文本中出现，那么它的IDF值应该低，而反过来如果一个词在比较少的文本中出现，那么它的IDF值应该高。

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer

tfidf=TfidfVectorizer()

tfidf.fit\_transform(corpus)

如上是对单词进行了tf-idf处理，处理之后单词转换为了特征，便可以此特征进行分类聚类处理。

需注意的是：在把测试样本进行TF-IDF处理由于训练样本已经训练好了模型只需tfidf.transform(testData)便可。在进行TF-IDF转化时它接收的参数是一个列表，**列表中的每一项是代表一个文档的字符串，不需要进行单词划分。**

例子：以Kaggle的比赛bag\_of\_wods为例

**TF-IDF进行情感分析**

*#encoding=utf-8***import** pandas **as** pd  
**import** numpy **as** np  
**import** xgboost **as** gbt  
**from** sklearn.feature\_extraction.text **import** TfidfVectorizer  
**from** sklearn.metrics **import** accuracy\_score  
**from** gensim.models **import** word2vec  
**from** sklearn.model\_selection **import** cross\_val\_score  
df=pd.read\_csv(**'labeledTrainData.tsv'**,sep=**'\t'**,header=0)  
label=df[**'sentiment'**].values  
training\_list=list(df[**'review'**])  
tfidf=TfidfVectorizer()  
X=tfidf.fit\_transform(training\_list)  
classer=gbt.XGBClassifier(n\_estimators=700,n\_jobs=-1)  
*#对*score=cross\_val\_score(classer,X,label,scoring=**'accuracy'**,cv=3)  
print(score.mean())  
*# classer.fit(X,label)  
# test\_data=pd.read\_csv('testData.tsv',sep='\t',header=0)  
# test\_list=list(test\_data['review'])  
# test\_X=tfidf.transform(test\_list)  
# yHat=classer.predict(test\_X)  
# submission = pd.DataFrame({  
# "id": test\_data["id"],  
# "sentiment": yHat  
# })  
# submission.to\_csv("bagofwords2-submission.csv", index=False)*

通过TF-IDF进行情感分析只是把文本粗粒度的分为正例与负例，且这种方法并没有考虑单词和上下文之间的联系，因此便出现了用深度学习的word2vec进行情感分析。

深度学习的word2vec进行情感分析

**word2vec是把词转换为词向量，同时能够结合上下文信息  
思路分析：  
1.一个样本作为一个列表中的一项，对每个样本进行分词组成行的列表  
，新列表中又有多个列表，每个列表存储的是分词后的一个样本（列表中是一个个单词）  
2.把单词向量化（一般训练样本较小时每个单词用100维表示）可得到所有单词的词向量  
3.得到每个单词的词向量之后需要得到一个文本的词向量表示具体为：  
把样本中各个单词的词向量进行相加然后除以样本中单词的个数，最后进行标准化处理  
4.经过步骤三处理数据就变成了样本个数行100列的数组便可进行分类处理了**

**注意它接收的数据是一个列表列表中元素是一个个单词，所以需要实现对文本进行单词划分**

df=pd.read\_csv(**'labeledTrainData.tsv'**,sep=**'\t'**,header=0)  
label=df[**'sentiment'**].values  
training\_list=list(df[**'review'**])  
*#对样本进行分词处理，这样才符合word2vec数据的接收格式*cut\_train\_data=[nltk.word\_tokenize(sen) **for** sen **in** training\_list]  
n\_dim=100 *#指定每个单词向量化后的维度  
#训练模型  
# model=word2vec.Word2Vec(cut\_train\_data,hs=1,min\_count=1,window=3,size=n\_dim)  
# model.save('bag\_model')*model=word2vec.Word2Vec.load(**'bag\_model'**)  
*#对已标注数据进行划分分为训练集和验证集*X\_train,X\_val,y\_train,y\_val=train\_test\_split(cut\_train\_data,label,test\_size=0.2)  
*#此函数依据词向量把样本转换为向量***def** buildWordVector(text,size):  
 vec=np.zeros(size).reshape((1,size))  
 count=0.  
 **for** word **in** text:  
 *#把样本中的每一个词向量进行相加* vec+=model[word].reshape((1,size))  
 count+=1.  
 **if** count!=0:  
 vec/=count  
 **return** vec  
*#按行链接每个样本*train\_vecs=np.concatenate([buildWordVector(sen,n\_dim) **for** sen **in** X\_train])  
*#进行标准化处理*train\_vecs=scale(train\_vecs)  
val\_vecs=np.concatenate([buildWordVector(sen\_val,n\_dim) **for** sen\_val **in** X\_val])  
val\_vecs=scale(val\_vecs)classer=gbt.XGBClassifier(n\_estimators=700,n\_jobs=-1)  
classer.fit(train\_vecs,y\_train)  
yHat=classer.predict(val\_vecs)  
print(accuracy\_score(yHat,y\_val))  
*# test\_data=pd.read\_csv('testData.tsv',sep='\t',header=0)  
# test\_list=list(test\_data['review'])  
# cut\_test\_data=[nltk.word\_tokenize(sen) for sen in test\_list]  
# model\_test=word2vec.Word2Vec(cut\_train\_data,hs=1,min\_count=1,window=3,size=100)  
# X\_test=model\_test.wv.vectors  
# yHat=classer.predict(X\_test)  
# submission = pd.DataFrame({  
# "id": test\_data["id"],  
# "sentiment": yHat  
# })  
# submission.to\_csv("bagofwords2-submission.csv", index=False)*

以上两者进行情感分析的区别

TF-IDT是建立词袋，获得每个单词的TF-IDF值，把所有样本中的每个单词作为一个特征，其特征数就是单词数，接收的数据是一个列表，列表中是一个个字符串代表一个样本，所以不需要就行单词划分。

Word2vec是把单词转化为向量，再用这些单词向量表示样本，用每个单词向量化后的列代表一个特征。接收的数据是一个列表，列表中的一个个列表代表一个样本，列表中是一个个单词，所以需要先进行单词划分。

文本主题分析LDA

详细参考http://www.cnblogs.com/pinard/p/6908150.html

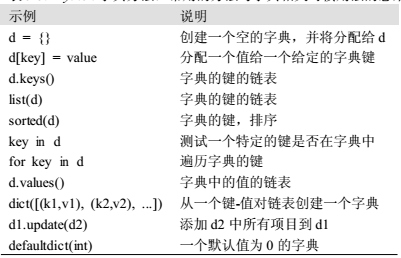
注意文本主题分析是基于词频向量的

27.Nltk中的字典

在nltk中可以创建带默认值的字典

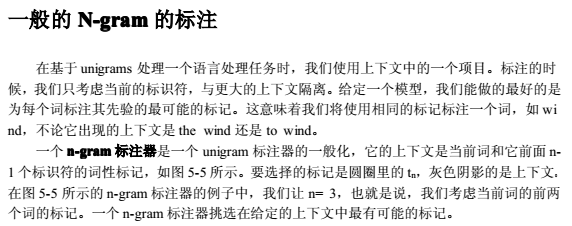
nltk.defaultdict(int)创建默认值为0的字典

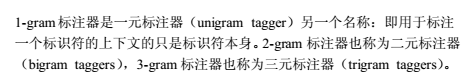
字典常用操作

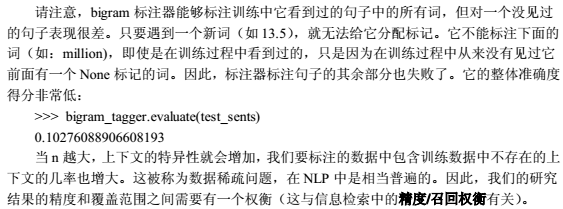


28.词性标注

1. gram标注







29.命名实体识别（named entity recognition,ner）

命名实体识别(Named Entity Recognition, NER)，又称作“专名识别”，主要任务是识别出文本中的人名、地名等专有名称和有意义的时间、日期等数量短语并加以归类。对很多文本挖掘任务来说，命名实体识别系统是重要的组成部分：一方面，命名实体识别可以帮助识别未登录词，而根据SIGHAN Bakeoff的数据评测结果，未登录词造成的分词精度损失远大于歧义；另一方面，对关键词提取等任务来说，命名实体的类别是非常有用的文本特征。

命名实体是命名实体识别的研究主体，一般包括３大类（实体类、时间类和数字类）和７小类（人名、地名、机构名、时间、日期、货币和百分比）命名实体。当然对于某些特定的应用场景，也可以把产品名、电影电视剧名、编程类库名等作为命名实体的类别。时间、日期、货币等实体识别通常可以采用模式匹配的方式获得较好的识别效果，而人名、地名、机构名的识别方法则比较复杂。

命名实体识别的过程通常分两步：识别实体边界、确定实体类别。英语中的命名实体具有比较明显的形态标志，如人名、地名等实体中的每个词的第一个字母要大写等，所以实体边界识别相对来说比较容易。中文内在的特殊性决定了在文本处理时首先必须进行词法分析，中文命名实体识别的难度要比英文的难度大。

使用stanfordcoreNLP进行ner以在python中下载模块还需一个jar包

具体参考：<https://blog.csdn.net/guolindonggld/article/details/72795022>

# 30.NLP之snowNLP和Textblob

Textblob主要用其对英文句子的情感分析

使用TextBlob情感分析的结果，以元组的方式进行返回，形式如(polarity, subjectivity). 其中polarity的分数是一个范围为 [-1.0 , 1.0 ] 浮点数, 正数表示积极，负数表示消极。subjectivity 是一个 范围为 [0.0 , 1.0 ] 的浮点数，其中 0.0 表示 客观，1.0表示主观的

**from** textblob **import** TextBlob

testimonial = TextBlob(**"Textblob is amazingly simple to use. What great fun!"**)  
*#返回的是textblob.en.sentiments.Sentiment类型的  
#第一个是polarity表示积极还是消极的，取值范围为[-1,1]，负数表示消极的正数表示积极的越接近两头表示程度越大  
#第二个是subjectivity表示主客观性，取值范围为[0,1]  
# print(testimonial.sentiment)  
# #也可以只输出表情绪的或主客观性的  
# print(testimonial.sentiment.polarity)  
# print(testimonial.sentiment.subjectivity)*

*#SnowNLP参照TextBlob主要用于对中文句子进行情感分析*text1=**'这个人脾气很好'**res=SnowNLP(text1)  
*#返回值为正面情绪的概率，越接近1表示正面情绪，越接近0表示负面情绪*print(res.sentiments)  
*#在进行句子情感分析之前他还可以训练自己的语料库*

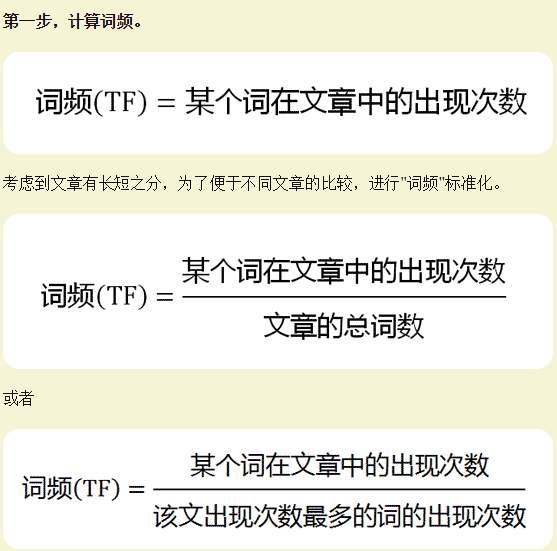
具体参考：

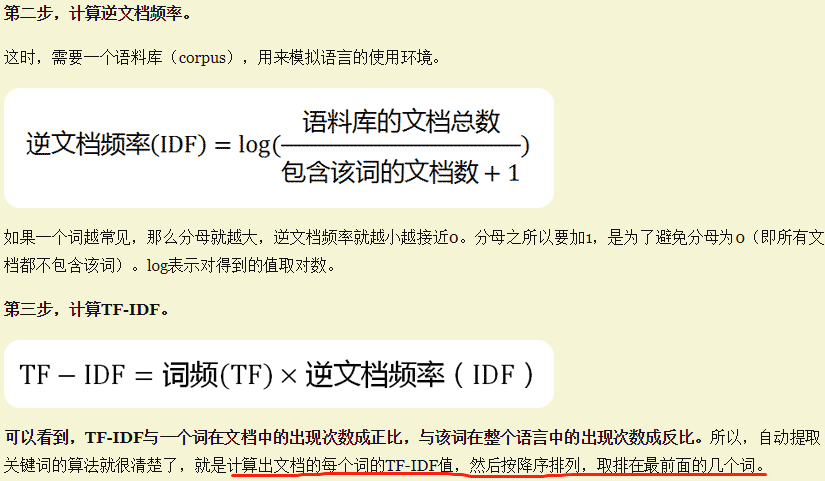
<https://zhuanlan.zhihu.com/p/26331196>

<https://blog.csdn.net/yibo492387/article/details/78783041>

31.文章关键词的提取

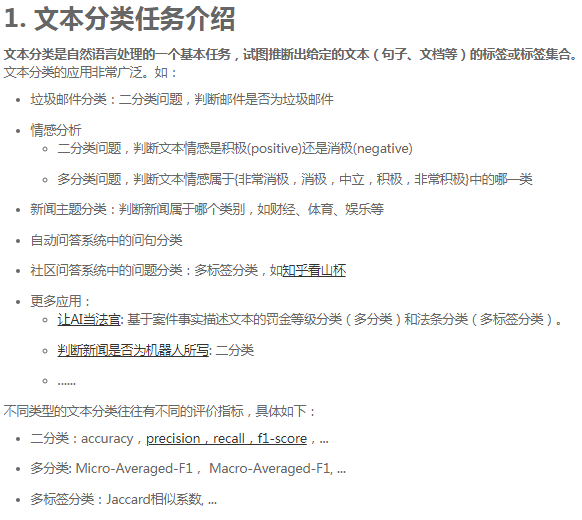
有个最简单的方法，利用TF-IDF效果不错

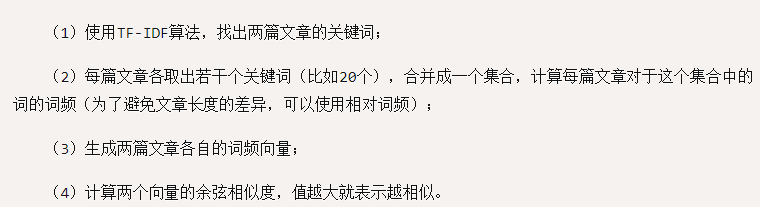




1. 利用TF-IDF与余弦相似度计算两个文章之间的相似度

文本分类的介绍





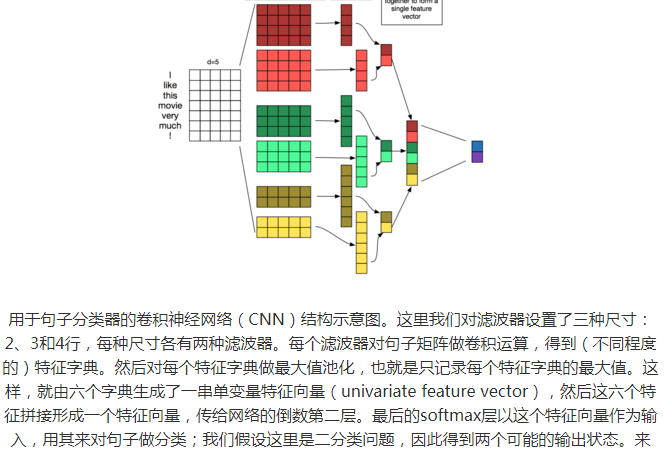
33.自动摘要的生成

http://www.ruanyifeng.com/blog/2013/03/automatic\_summarization.html

34.Text-CNN

用简单的CNN对文本进行分类

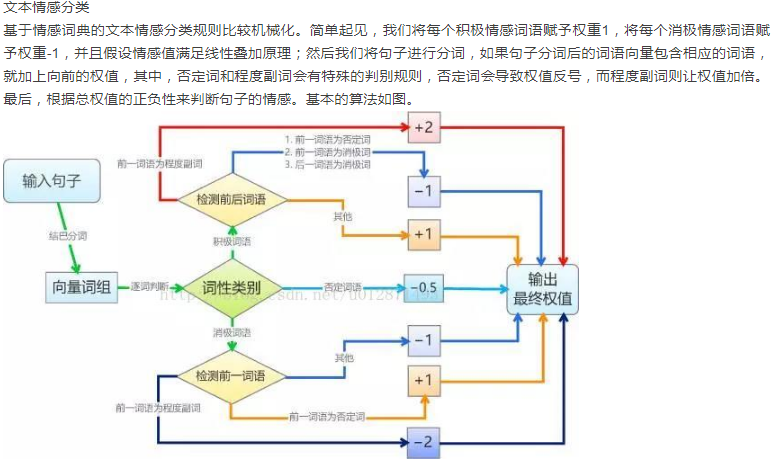
它用的结构比较简单，就是使用长度不同的filter对文本矩阵进行卷积，filter的宽度等于词向量的长度，然后使用max-pooling对每一个filter提取的向量进行操作，最后每一个filter对应一个数字，把这些filter拼接起来，就得到了一个表征该句子的向量，最后的预测都是基于该句子的。



35.粗粒度情感分析

所谓粗粒度情感分析只是把文本整体简单的标记为积极的或者消极的。

传统方法有一种基于情感词典的情感分析，需要实现准确情感词典，

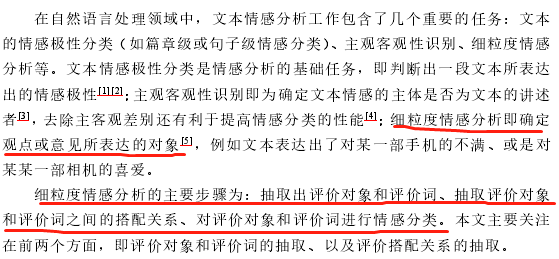


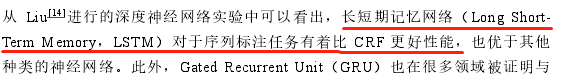
以上方法存在缺点，因为基于情感词典的文本情感分类对于词典的要求很高，会有很多新词或者词典中不存在的词，因此需要对情感词典进行扩充。方法是：获取相关领域的大量数据，基于已有的词典构建的模型对文本进行分类，然后统计单词的词频，对于那些在积极文本中出现次数很多在消极文本中出现很少的词可以加入积极词典，同理也可以加入消极词典，词典扩充后再重新对文本进行分类。

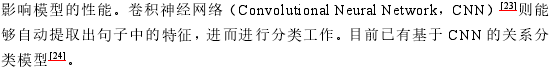
**以上模型有很大的缺点，需要相关领域知识强，词典中词语的情感需要相关专家进行标注很耗费人力。**

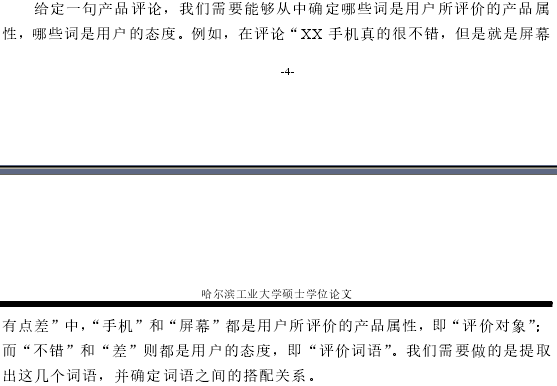
36.细粒度情感分析

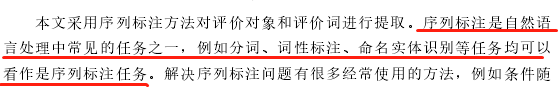




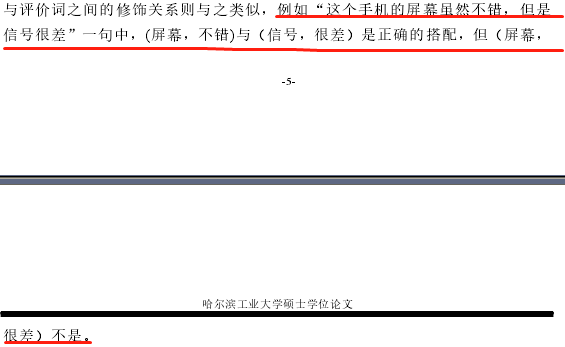


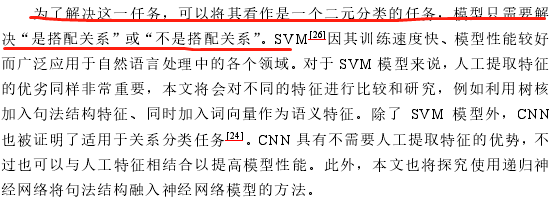






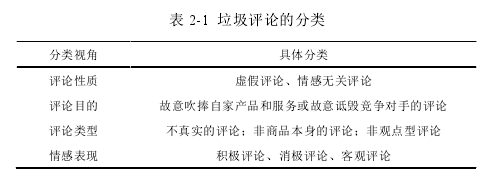


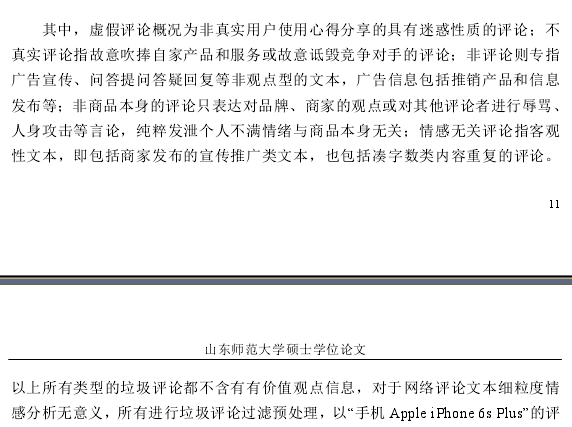




**另一篇论文**

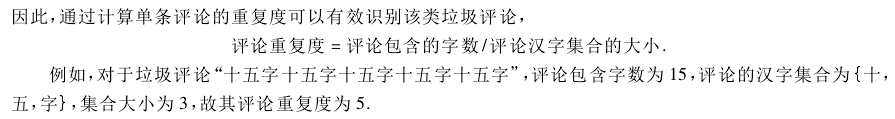
第一、去除垃圾评论





具体应该结合数据集，把其转换为一个二分类问题，去除垃圾评论。（如何区分垃圾评论？）

1）评论重复度高的去除



1. 重复的评论去除

即为已存在与其相同的评论应该去除

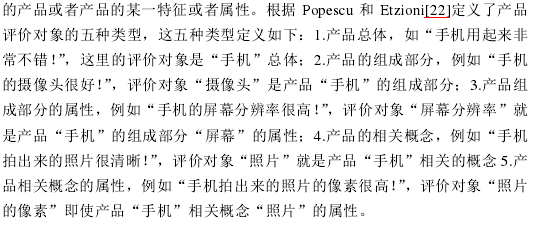
1. 去除存在微信，QQ，电话以及超链接的垃圾评论（利用正则表达式）
2. 去除具备网络低俗语的评论(要收集网络低俗语)
3. 去除错评的数据
4. 把不含评价词的评价直接定义为中性



通过以上五条可以直接去除一些样本，剩余的样本应该转换为二分类问题，所以要进行特征的提取。

1. 对于有效评论而言，含有主题词（即为描述主题属性的词，例如对于收集的评论，外观，电池等）很重要，所以可以把分词后一个句子主题词所占的比例作为特征项
2. 评价词（能代表情感属性的形容词，好看等），可以把分词后一个句子评价词所占的比例作为特征项
3. 广告词是垃圾评论的标志，可以把分词后一个句子广告词所占的比例作为特征项
4. 评论者的打分
5. 评论的有效性得票数
6. 进行词性标注，主要是选取评论中的名词，以及名词短语（情感对象）和形容词（情感要素），对于没有情感词的一般理解为中性评论。

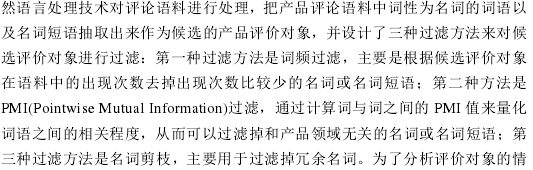
情感对象的类别如下



**显式情感对象的抽取方法如下：**

进行词性标注，主要是选取评论中的名词，以及名词短语（情感对象）

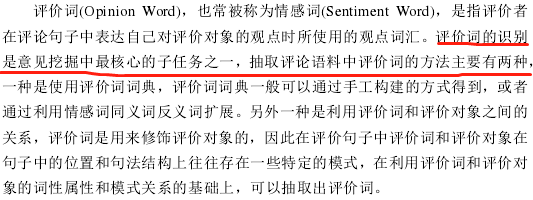
**随后还要进行过滤**,方法如下



这种方法找到的只是被大多数人所关注的频繁出现的情感对象（名词），但是有些情感对象不是被人们所广泛关注的，可以利用评价词来找出这些不是频繁出现的评价词（具体为对于一个不包含频繁出现的评价对象的句子，找该句子中离评价词最近的名词作为评价对象。）那么把名词全选出来不过滤会如何？

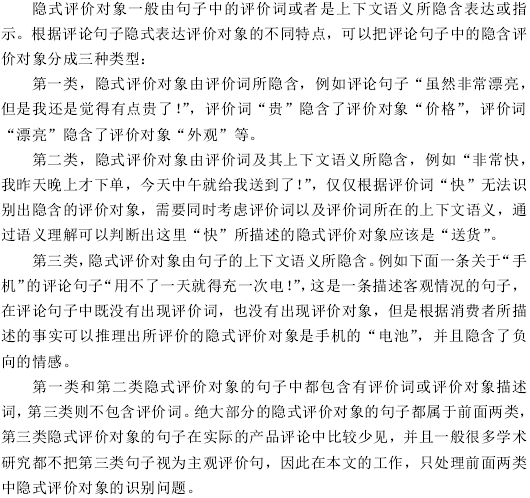
**评价词的抽取方法如下：**

选取形容词（情感要素）作为评价词



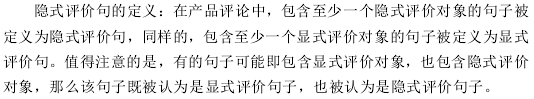
**隐式情感对象的抽取：**

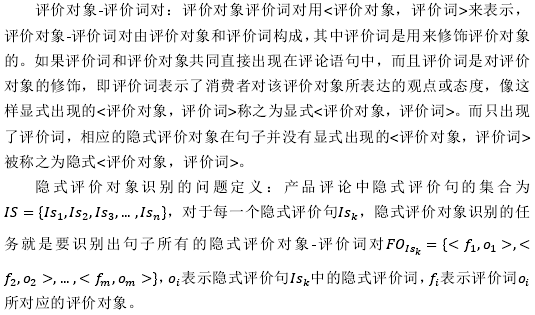
**隐式情感对象的类别：**



但是由于评论的口语化以及随意性，评论的大多数往往没有情感对象（名词），这用到了隐式情感对象的识别，需要结合上下文语义，在这里我们可以把其转换为分类问题。(如何识别隐式情感对象？论文中用朴素贝叶斯分类器)

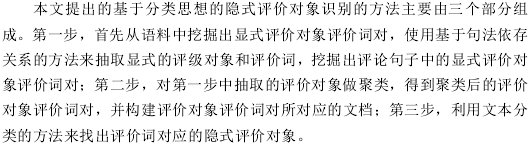
隐式评价句的定义：





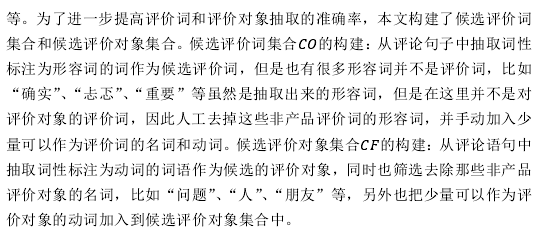
那么隐式句子如何识别？可以基于句法依存关系，若对评论做完句法依存关系还有不在SVB以及DE这两类关系中，但是在候选评价词集中的评价词存在，说明它无对应的评价对象则为隐式评价句。

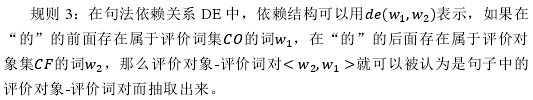
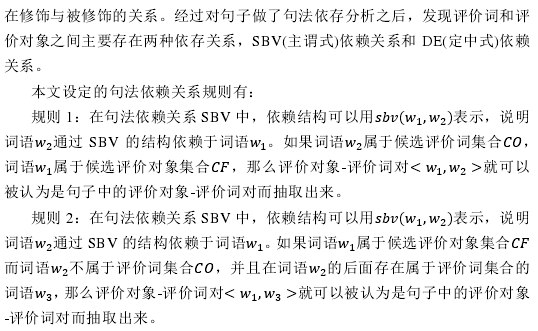
1）先构建<情感对象,情感要素>对，对于那些隐性的情感对象可以根据前面构建的对进行补充。（这里牵扯到一个问题，若某句评价中没有显示的评价对象，同时其评价语也没在构建的对中出现怎么办？这就要先对评价对象，评价词进行聚类，对每一类选取质心作为代表进行对的构建，其余的通过簇的映射进行，这时对于这个评价词可以先找其所属簇，看与其所属簇质心对应的评价对象，即为其描述的评价对象）



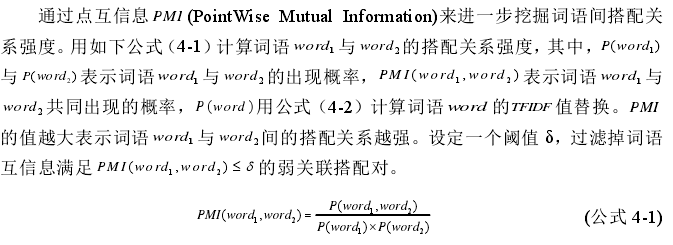
**另一种方法**：利用依存句法分析的SVB（即主谓关系），一般主语为情感对象，谓语为情感要素。或者DE（定中式）依赖关系。后面的为评价对象，前面的为评价语。也用上面方法进行过滤。

**整个的实施流程：**

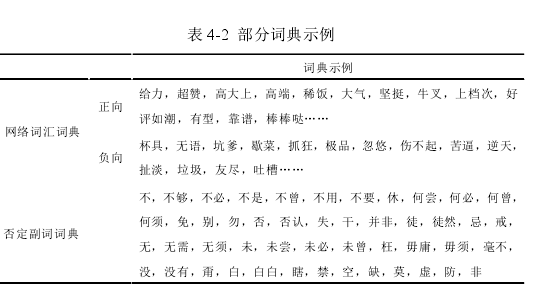


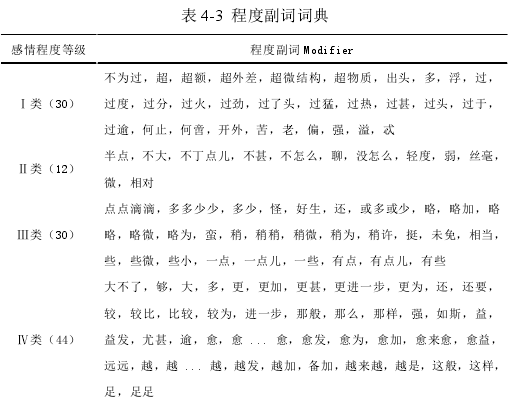


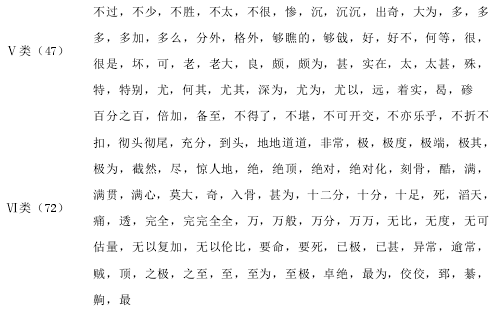
1. 之所以在第二步要进行词性的标注，是要组成<情感对象，情感要素>这样的词对，用以描述用户对某一产品每个属性的具体评价。例如<音质,好>,<用电,快>等等。但是由于网络评论的随意性，情感对象的表达往往是多样的，也就是不同的表述可能描述的是同一产品属性，例如，论文中采取的方法是把单词转换为词向量，再对单词进行聚类，以每一簇的质心作为这一类情感对象的代表，并赋予每一类一个情感代码。（用word2vec进行词向量转换再进行聚类是否可行？）
2. 选取了情感对象与情感要素随后还要考虑搭配问题，论文中采用频繁项集但是效果不好，因此引用了PMI进行评定



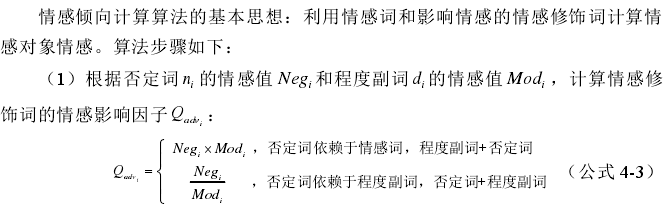
1. 最终的目的是构造<情感对象、情感要素、情感强度>这样的记录形式，前面的任务<情感对象、情感要素>的构造已经完成，随后应该设计算法计算评论的情感强度。

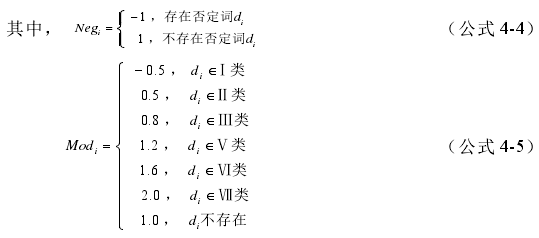


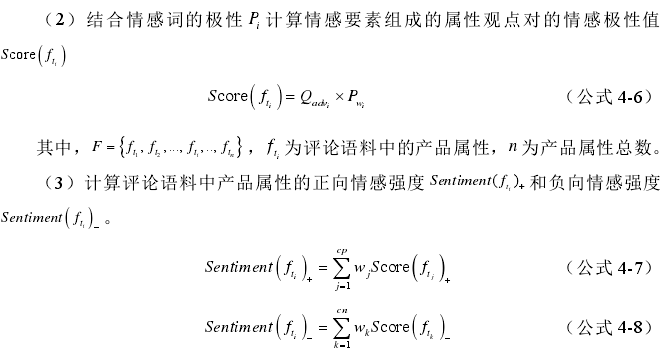


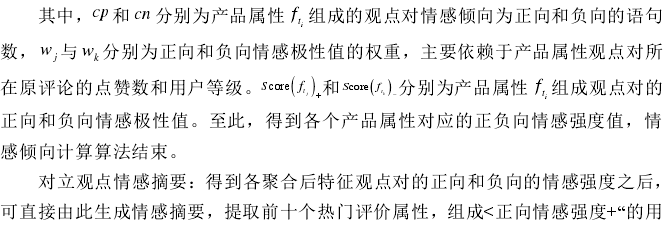


正负向情感强度分析算法











**最后整体的总结**

获取评论语料，每一个评论作为一个文档。第一、删除垃圾评论：评论语料中的评论并不是都能用，可以去除一些垃圾评论。第二、构造评价对象以及评价词候选集合：进行词性标注，大多数情况而言评价对象一般为名词或名词短语，评价词一般为形容词，但是也有不是这种情况的。为了增加准确度，可以构建评价对象集合，手动的对其进行处理（包括删除不是评价对象的名词以及名词短语，添加相关领域涉及的不是名词或名词短语的评价对象），构架评价词集合，手动的做与评价对象集合相同的操作。对于那些不包含评价词候选集合中评价词的评价语料可以标注为中性。第三、利用句法依存分析构建显式的<评价对象，评价词>对。具体做法参考上面介绍（对于对还要进行过滤参考上面）。第四、为隐式句匹配评价对象。（对于隐式句的定义参考上面。）这里有一个问题，如何判别何为隐式句，可以基于句法依存关系，若对评论做完句法依存关系还有不在SVB以及DE这两类关系中，但是在候选评价词集中的评价词存在，说明它无对应的评价对象则为隐式评价句。其次如何为隐式句匹配评价对象？可以利用聚类算法，把评价词和评价对象用

word2vec转换为词向量，对评价词和评价对象进行聚类，找到隐式评价句中评价词所属的簇，（这里的想法，一把显式句评价词进行聚类隐式句加入离其最近的质心中，这样就可以找隐式评价词所在簇质心这个评价词的评价对象作为其评价对象。二、聚类后用k近邻进一步进行投票表决）

37.依存句法分析与语义依存分析的区别

依存语法 (Dependency Parsing, DP) 通过分析语言单位内成分之间的依存关系揭示其句法结构。 直观来讲，依存句法分析识别句子中的“主谓宾”、“定状补”这些语法成分，并分析各成分之间的关系。

语义依存分析 (Semantic Dependency Parsing, SDP)，分析句子各个语言单位之间的语义关联，并将语义关联以依存结构呈现。 使用语义依存刻画句子语义，好处在于不需要去抽象词汇本身，而是通过词汇所承受的语义框架来描述该词汇，而论元的数目相对词汇来说数量总是少了很多的。语义依存分析目标是跨越句子表层句法结构的束缚，直接获取深层的语义信息。

具体参考：<http://www.cnblogs.com/CheeseZH/p/5768389.html>

38.TextRank进行关键词，关键短语提取以及摘要生成

将原文本拆分为句子，在每个句子中过滤掉停用词（可选），并只保留指定词性的单词（可选）。由此可以得到句子的集合和单词的集合。

每个单词作为pagerank中的一个节点。设定窗口大小为kk，假设一个句子依次由下面的单词组成：

w1,w2,w3,w4,w5,…,wnw1,w2,w3,w4,w5,…,wn

[w1,w2,…,wkw1,w2,…,wk]、[w2,w3,…,wk+1w2,w3,…,wk+1]、[w3,w4,…,wk+2w3,w4,…,wk+2]等都是一个窗口。在一个窗口中的任两个单词对应的节点之间存在一个无向无权的边。

基于上面构成图，可以计算出每个单词节点的重要性。最重要的若干单词可以作为关键词。

具体参考：<https://www.jianshu.com/p/1755d79282a1以及连接的博文>

使用参考：guoji\_anquan\_shijian下的guanjianci\_tiqu.py