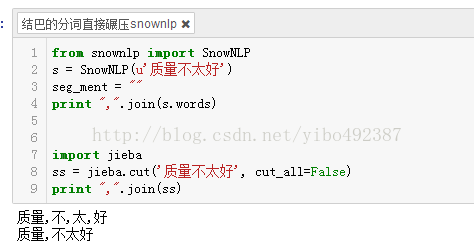
今天逛网页，无意中看到了有人写[snownlp](https://so.csdn.net/so/search?q=snownlp&spm=1001.2101.3001.7020)这个库用来处理文本的情感，出于好奇，出于安慰吧（因为这两天想装hanlp，结果没装上，总是报错，心情巨沮丧！！，所以逛着其他包替代hanlp）

发现jieba的[分词](https://so.csdn.net/so/search?q=%E5%88%86%E8%AF%8D&spm=1001.2101.3001.7020)直接碾压snownlp的分词，snownlp的情感分析也是巨慢，其中把训练中包括了分词部分，巨慢，不可用的赶脚！

举例如下：

更多的例子就不再一一列举了！

不过还是把介绍snownlp的文章贴在下面

http://blog.csdn.net/yyxyyx10/article/details/62428238

**python的中文**[**文本**](https://so.csdn.net/so/search?q=%E6%96%87%E6%9C%AC&spm=1001.2101.3001.7020)**挖掘库snownlp进行购物评论文本情感分析实例**

 昨晚上发现了snownlp这个库，很开心。先说说我开心的原因。我本科毕业设计做的是文本挖掘，用R语言做的，发现R语言对文本处理特别不友好，没有很多强大的库，特别是针对中文文本的，加上那时候还没有学机器学习算法。所以很头疼，后来不得已用了一个可视化的软件RostCM，但是一般可视化软件最大的缺点是无法调参，很死板，准确率并不高。现在研一，机器学习算法学完以后，又想起来要继续学习文本挖掘了。所以前半个月开始了用python进行文本挖掘的学习，很多人都推荐我从《python自然语言处理》这本书入门，学习了半个月以后，可能本科毕业设计的时候有些基础了，再看这个感觉没太多进步，并且这里通篇将nltk库进行英文文本挖掘的，英文文本挖掘跟中文是有很大差别的，或者说学完英文文本挖掘，再做中文的，也是完全懵逼的。所以我停了下来，觉得太没效率了。然后我在网上查找关于python如何进行中文文本挖掘的文章，最后找到了snownlp这个库，这个库是国人自己开发的python类库，专门针对中文文本进行挖掘，里面已经有了算法，需要自己调用函数，根据不同的文本构建语料库就可以，真的太方便了。我只介绍一下这个库具体应用，不介绍其中的有关算法原理，因为算法原理可以自己去学习。因为我在学习这个库的时候，我查了很多资料发现很少或者基本没有写这个库的实例应用，很多都是转载官网对这个库的简介，所以我记录一下我今天的学习。

        首先简单介绍一下这个库可以进行哪些文本挖掘。snownlp主要可以进行中文分词（算法是Character-Based Generative Model）、词性标注（原理是TnT、3-gram 隐马）、情感分析（官网木有介绍原理，但是指明购物类的评论的准确率较高，其实是因为它的语料库主要是购物方面的，可以自己构建相关领域语料库，替换原来的，准确率也挺不错的）、文本分类（原理是朴素贝叶斯）、转换拼音、繁体转简体、提取文本关键词（原理是TextRank）、提取摘要（原理是TextRank）、分割句子、文本相似（原理是BM25）。官网还有更多关于该库的介绍，在看我这个文章之前，建议先看一下官网，里面有最基础的一些命令的介绍。官网链接：https://pypi.python.org/pypi/snownlp/0.11.1。

        下面正式介绍实例应用。主要是中文文本的情感分析，我今天从京东网站采集了249条关于笔记本的评论文本作为练习数据，由于我只是想练习一下，没采集更多。然后人工标注每条评论的情感正负性，情感正负性就是指该条评论代表了评论者的何种态度，是褒义还是贬义。以下是样例



其中-1表示贬义，1表示褒义。由于snownlp全部是unicode编码，所以要注意数据是否为unicode编码。因为是unicode编码，所以不需要去除中文文本里面含有的英文，因为都会被转码成统一的编码（补充一下，关于编码问题，我还是不特别清楚，所以这里不多讲，还请对这方面比较熟悉的伙伴多多指教）。软件本身默认的是Ascii编码，所以第一步先设置软件的默认编码为utf-8，代码如下：

1、改变软件默认编码

import sys  
reload(sys)  
sys.setdefaultencoding('utf-8')

2、然后准备数据

import pandas as pd #加载pandas  
text=pd.read\_excel(u'F:/自然语言处理/评论文本.xlsx',header=0) #读取文本数据  
text0=text.iloc[:,0] #提取所有数据  
text1=[i.decode('utf-8') for i in text0] #上一步提取数据不是字符而是object，所以在这一步进行转码为字符

3、训练语料库

from snownlp import sentiment #加载情感分析模块  
sentiment.train('E:/Anaconda2/Lib/site-packages/snownlp/sentiment/neg.txt', 'E:/Anaconda2/Lib/site-packages/snownlp/sentiment/pos.txt') #对语料库进行训练，把路径改成相应的位置。我这次练习并没有构建语料库，用了默认的，所以把路径写到了sentiment模块下。  
sentiment.save('D:/pyscript/sentiment.marshal')#这一步是对上一步的训练结果进行保存，如果以后语料库没有改变，下次不用再进行训练，直接使用就可以了，所以一定要保存，保存位置可以自己决定，但是要把`snownlp/seg/\_\_init\_\_.py`里的`data\_path`也改成你保存的位置，不然下次使用还是默认的。

4、进行预测

from snownlp import SnowNLP  
senti=[SnowNLP(i).sentiments for i in text1] #遍历每条评论进行预测

5、进行验证准确率

预测结果为positive的概率，positive的概率大于等于0.6，我认为可以判断为积极情感，小于0.6的判断为消极情感。所以以下将概率大于等于0.6的评论标签赋为1，小于0.6的评论标签赋为-1，方便后面与实际标签进行比较。

newsenti=[]  
for i in senti:  
  if (i>=0.6):  
      newsenti.append(1)  
  else:  
      newsenti.append(-1)  
text['predict']=newsenti  #将新的预测标签增加为text的某一列，所以现在text的第0列为评论文本，第1列为实际标签，第2列为预测标签

counts=0  
for j in range(len(text.iloc[:,0])): #遍历所有标签，将预测标签和实际标签进行比较，相同则判断正确。  
    if text.iloc[j,2]==text.iloc[j,1]:  
        counts+=1  
print u"准确率为:%f"%(float(counts)/float(len(text)))#输出本次预测的准确率

运行结果为：



准确率还可以，但还不算高，原因是我考虑时间原因，并且我只是练习一下，所以没有自己构建该领域的语料库，如果构建了相关语料库，替换默认语料库，准确率会高很多。所以语料库是非常关键的，如果要正式进行文本挖掘，建议要构建自己的语料库。在没有构建新的语料库的情况下，这个83.9357%的准确率还是不错了。

         以上是我这次的学习笔记，和大家分享一下，有不足之处请大家批评指正。我还是一个刚涉世数据挖掘、机器学习、文本挖掘领域不久的小白，有许多知识还是比较模糊，但对这数据挖掘很感兴趣。希望能多结识这方面的朋友，共同学习、共同进步