**华中科技大学计算机科学与技术学院**

**《机器学习》课堂二结课报告**



专 业： 计算机科学与技术

班 级： CS1904

学 号： U201915034

姓 名： 龚宇蒙

成 绩：

指导教师： 黄宏

**完成日期： 2021年 11 月 27 日**

目录

[新闻数据情感分析 3](#_Toc27938)

[一、 实验题目：新闻数据情感分析 3](#_Toc6670)

[二、实验要求 3](#_Toc31762)

[2.1题目背景 3](#_Toc1944)

[2.2数据集 3](#_Toc19586)

[2.3任务描述 3](#_Toc4800)

[2.4评测标准 3](#_Toc24259)

[三、算法设计 4](#_Toc17781)

[3.1数据处理 4](#_Toc31844)

[3.2模型算法 4](#_Toc29944)

[3.3模型评估 5](#_Toc22472)

[四、实验环境与平台 5](#_Toc7076)

[五、 程序实现 6](#_Toc21849)

[5.1工程文件目录结构 6](#_Toc31181)

[5.2数据集清洗翻译代码 6](#_Toc13160)

[5.3合并json文件 8](#_Toc32420)

[5.4 贝叶斯模型预测结果 9](#_Toc11769)

[六、 实验结果 11](#_Toc16716)

[6.1数据集翻译结果 11](#_Toc26127)

[6.2贝叶斯分类模型预测结果 11](#_Toc21377)

[6.3 educoder测试结果 12](#_Toc20530)

[七、 结果分析 12](#_Toc17017)

[7.1输出结果 12](#_Toc19523)

[7.2算法改进和调参优化 13](#_Toc32757)

[八、 机器学习的课程体会和建议 14](#_Toc22751)

[8.1 结课实验总结 14](#_Toc31889)

[8.2课程总结 15](#_Toc9946)

# 新闻数据情感分析

## 实验题目：新闻数据情感分析

## 二、实验要求

### 2.1题目背景

随着各种社交平台的兴起，网络上用户的生成内容越来越多，产生大量的文本信息，如新闻、微博、博客等，面对如此庞大且富有情绪表达的文本信息，完全可以考虑通过探索他们潜在的价值为人们服务。因此近年来情绪分析受到计算机语言学领域研究者们的密切关注，成为一项进本的热点研究任务。

本赛题目标为在庞大的数据集中精准的区分文本的情感极性，情感分为正、负两类。面对浩如烟海的新闻信息，精确识别蕴藏在其中的情感倾向，对舆情有效监控、预警及疏导，对舆情生态系统的良性发展有着重要的意义。

### 2.2数据集

本实验使用dataset数据集用于本次新闻情感分析。数据集分为训练集和测试集两部分。训练集train.json共有2200篇新闻组文档数据，测试集test.json共有907篇新闻组文档数据。

### 2.3任务描述

本次实验需要对我们提供的新闻数据进行情感极性分类，其中新闻内容客观对应1，新闻内容不客观对应0。根据我们提供的训练数据，通过您的算法或模型判断出测试集中新闻的情感极性。

注意标签只有0和1，即本次实验一个二分类问题。

### 2.4评测标准

学生将预测得到的测试集中新闻的情感极性标签label复制至educoder实训平台中，得到预测结果的micro-f1和marco-f1值。

## 三、算法设计

### 3.1数据处理

本次实验所给的数据集语言混杂，包含了各种新闻语言。因此我们首先需要对于实验所给的train.json以及test.json中的数据进行清洗，将其统一成一种语言，否则将无法使用朴素贝叶斯模型算法对数据分析预测。

对于自然语言的翻译，需要调用API。可选的有百度翻译、有道翻译以及Google翻译。由于百度翻译api需要注册并且需要密钥，所以这里数据处理我们选择使用Google翻译的google\_trans\_new库，他是谷歌翻译库Googletrans的替代品。后者由于科学上网的原因如今国内无法使用(国内需要使用节点代理)。

通过使用google翻译api，将原数据集进行清洗翻译，得到了全部翻译成英文的处理后的数据集train\_translated.json和test\_translated.json。之后采用朴素贝叶斯模型对本次数据集进行分析和预测，最后为测试集打上预测的label标签结果。

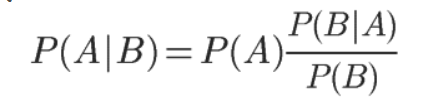
### 3.2模型算法

本次实验使用的模型算法为朴素贝叶斯算法，调用了sklearn朴素贝叶斯实现API。

朴素[贝叶斯](https://so.csdn.net/so/search?from=pc_blog_highlight&q=%E8%B4%9D%E5%8F%B6%E6%96%AF" \t "https://blog.csdn.net/houyanhua1/article/details/_blank)预测分类的思想就是根据待预测文档的特征(TF-IDF高的词)分别计算属于各个类别的概率，其中概率最大的类别，就是预测的类别。（朴素的意思就是文档的特征(词)之间相互独立）

贝叶斯推断：

对条件概率公式进行变形，可以得到如下形式：



P(A)称为"先验概率"（Prior probability），即在B事件发生之前，对A事件概率的一个判断。

P(A|B)称为"后验概率"（Posterior probability），即在B事件发生之后，对A事件概率的重新评估。

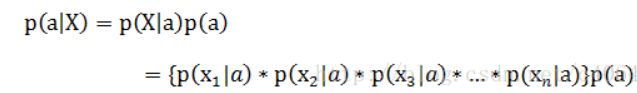
P(B|A)/P(B)称为"可能性函数"（Likelyhood），这是一个调整因子，使得预估概率更接近真实概率。

朴素贝叶斯推断:

我们来看一看朴素贝叶斯推断，对比之前讲过的贝叶斯推断，你可以发现贝叶斯和朴素贝叶斯的概念是不同的，区别就在于“朴素”二字，朴素贝叶斯对条件概率分布做了条件独立性的假设。 比如下面的公式，假设有n个特征：



由于每个特征都是独立的，我们可以进一步拆分公式：



这就是贝叶斯分类器的基本方法：在统计资料的基础上，依据某些特征，计算各个类别的概率，从而实现分类。

### 3.3模型评估

对于新闻情感分析这样的二分类贴标签问题，采用贝叶斯分类算法还是非常方便的。朴素贝叶斯模型发源于古典数学理论，有稳定的分类效率。同时对缺失数据不太敏感(缺词少句)，算法也比较简单,而且分类准确度高，速度快。

对于本次实验的数据集，相比于K近邻和决策树采用朴素贝叶斯算法是方便的。但是朴素贝叶斯算法也具有许多缺点:1.需要知道先验概率P(F1,F2,…|C)，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。(要求特征词之间相互独立)2.受训练集的误差(特征词的提取，例如某篇文章有大量相同的词)影响较大。

## 四、实验环境与平台

实验环境：Window 10系统

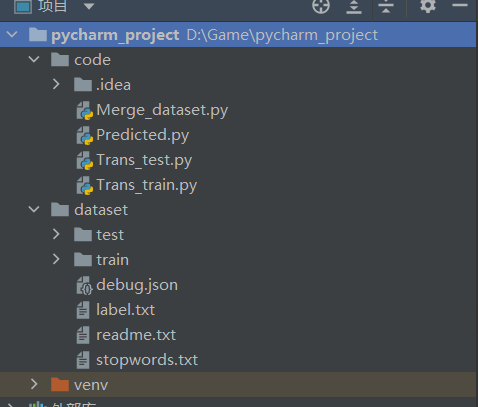
软件平台：PyCharm2021.2.3

Python解释器版本:Python 3.8

## 程序实现

### 5.1工程文件目录结构

本实验的工程代码结构如下图所示:



其中Trans\_test.py和Trans\_train.py分别实现了对于数据集test.json以及train.json的翻译清洗，而Merge\_dataset.py则是将分割之后的数据集重新组合合成一个json文件的过程。Predicted.py实现利用朴素贝叶斯模型对于测试集的分析和预测过程。最终生成的label标签列表数据通过文件读写写入label.txt。

Dataset文件夹中的readme.txt 文本中描述了对于train.json和test.json原始数据集的切割过程。

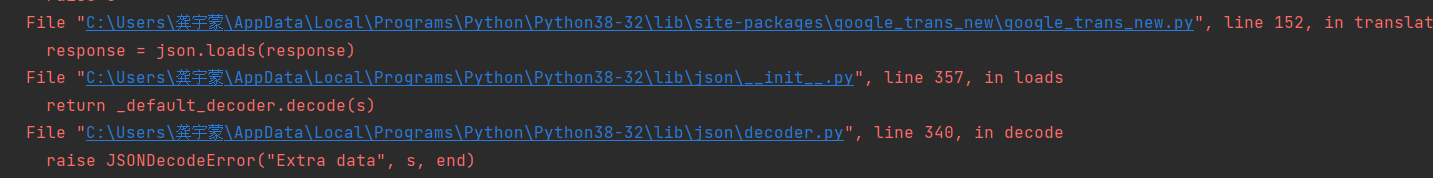
### 5.2数据集清洗翻译代码

本次实验很重要的一个过程就是对于数据集进行清洗。用pycharm查看了原json文件之后，可以发现原数据集语言体系很混乱，各种语言类型都有。为了方便我们后面对数据集采用模型处理，需要先对数据集进行清洗，实现自然语言翻译。这里统一将数据集内容翻译成英文。

值得注意的一件事是，train.json文件内容和test.json文件内容还不尽相同。查看后可以发现，train.json每条数据都是一个字典的形式。即完整的content 和label这样的键值对。而test.json中数据只有content而没有label。这也是很正常的事情，我们的目标就是预测得到test.json的label列表，即所谓的贴标签过程。

因此我们对于数据集的处理就需要分类了。所以这里使用Trans\_train.py来处理train.json文件，使用Trans\_test.py处理test.json文件。

对于数据的翻译过程，我们需要使用到google翻译API。Pycharm非常方便便的为我们提供了google\_trans\_new这个软件包，可以直接pip install之后就可以使用了。最开始的时候我使用的是Googletrans这个包来作为翻译工具，但是因为国内代理的问题，每次翻译一会就会出现中断的情况(google服务器会检测到异常流量)，所以我选择了这个包作为替代。当然这个包也是具有一个小bug的。最开始我直接运行翻译程序的时候，会出现中断情况如下图：

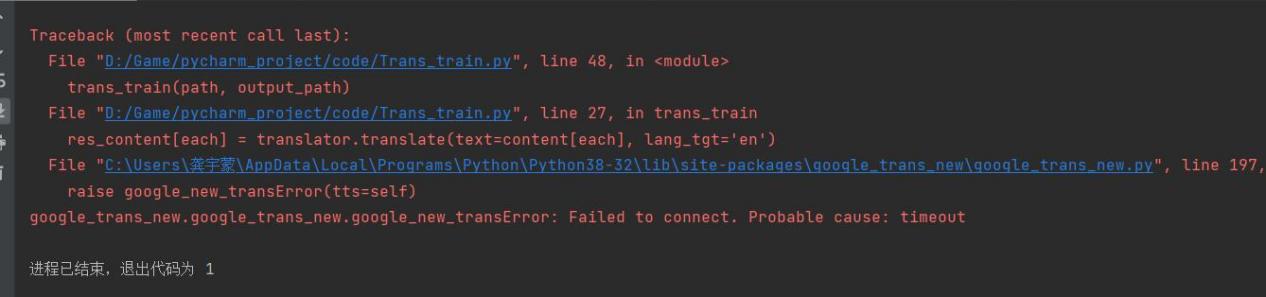


这个问题经过上网查阅后得到了解答:



这里我们采用第一个方法，点开google\_trans\_new.py,找到151行进行修改。修改之后便可以正常运行了。

当然运行的时候也经常遇到这种情况：



这里出现了超时的问题。由于translator默认设置的翻译超时时间是5s,但是我们实际中可能出现翻译较大数据的情况，可能5s确实还在翻译内容，这里我们手动将translator超时时间设置为10s。

使用google\_trans\_new库时，存在另一个问题就是一次翻译的内容问题。Google翻译一次允许翻译的字符数目要求不超过5000字，超过5000字会自动返回异常。但是我们需要翻译的文本内容很多就超过了5000字。因此我们在读取数据集文本内容的时候，需要判断此条数据content内容是否超过5000，超过5000我们需要将此content进行字符串分割存入一个列表Cuted\_content中，再对列表的成员进行翻译，翻译完成后再将成员合并存入res\_content中。

所以对于train.json文件，翻译的代码主要实现以下步骤：

1. 打开train.json文件，分别将每一个条数据的content和label存入列表中
2. 判断每条数据的content长度。如果长度小于等于5000，则直接调用同translator.translate()对content进行翻译。如果长度大于5000，利用textwrap.wrap()

函数将content分割成长度为3000的字符串列表，然后对于列表的每个成员调用translator.translate()翻译。

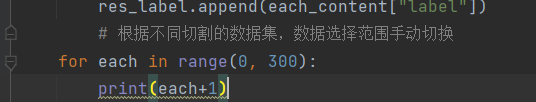
1. 将翻译后的content和label重新存入字典temp中。
2. 将字典放入列表translated\_json中。
3. 打开输出的文件路径，用json.dump(translated\_json,f)将翻译之后的数据写入json文件。

对于test.json文件，翻译的代码步骤基本和上面相同。唯一的不同之处在第

1步仅将content存入列表，第3步仅将content存入字典temp之中。

开始的时候我想直接一次对train.json的2200条数据直接翻译。但是这样总会出现翻译待机太久中断的情况。因此想到对于数据集进行划分，分来处理。每次翻译300-500条数据，先读入json文件。将所有的数据翻译之后，再将得到的json文件的内容合并到一个json文件当中。这对于python实现来说时非常简单的。

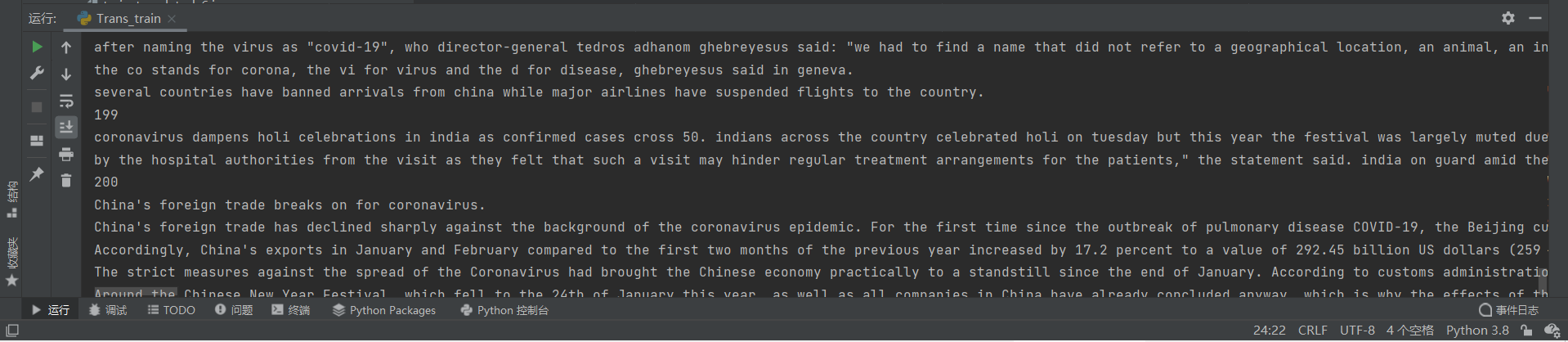
因此我选择将train.json的2200条数据划分6次翻译处理，test.json的907条数据划分2次翻译处理。每次仅需要修改Trans\_train.py和Trans\_test.py的以下代码段：



需要处理第i条-第j条数据时，就将range修改成range(i,j+1)即可。

处理完所有数据集，得到了train\_translated\_1.json-train\_translated\_6.json以及test\_translated\_1.json-test\_translated\_2.json。接下来需要做的就是将这些json文件进行合并操作。

每次翻译数据时的控制台输出结果如图：



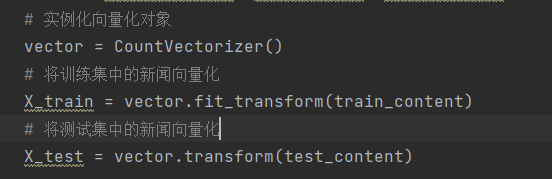
### 5.3合并json文件

上面我们通过翻译处理得到了多个分割的翻译后的json数据文件。现在通过Merge\_dataset.py合并数据集。Json软件包提供了很方便的对json文件的处理。打开每个文件，通过load()读取文件数据到data中,之后利用append将数据加载到train\_content和test\_content列表中，最后通过dump()将合并后的数据放入一个json文件中。

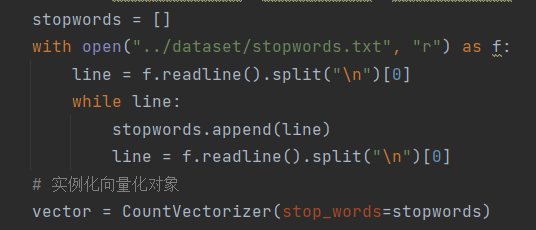
### 5.4 贝叶斯模型预测结果

有了处理好的数据集，接下来就是使用选择的模型对数据集进行处理。我们选择的是朴素贝叶斯模型，通过调用sklearn库来使用sklearn.naive\_bayes.MultinomialNB()函数处理。由于数据集中每一条数据都是很长的一个字符串，所以我们需要对数据进行向量化的处理。例如，I have a apple！ I have a pen!可能需要将该字符串转换成向量如[10, 7, 0, 1, 2, 6, 22, 100, 8, 0, 1, 0]。

sklearn库提供了实现词频向量化功能的CountVectorizer类。想要对数据进行向量化，代码如图：

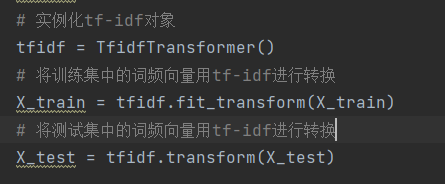


观察json文件中content可知，content为新闻文本信息。可以想到会出现很多无用的常用词，所以考虑到用停用词处理。根据我搜寻资料后，在网上找到了当前最全的英文停用词表。读取此文件将停用词读入停用词列表stopwords,之后再将停用词列表传入实例化函数中。如下图：



但是仅仅通过统计词频的方式来将文本转换成向量会出现一个问题：长的文章词语出现的次数会比短的文章要多，而实际上两篇文章可能谈论的都是同一个主题。

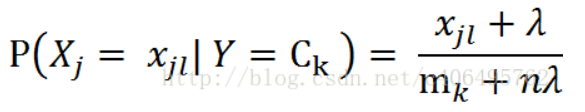
为了解决这个问题，我们可以使用tf-idf来构建文本向量，sklearn中已经提供了tf-idf的接口，具体实现过程如下：



接下来就需要用到朴素贝叶斯模型分析预测了。这也是整个机器学习实验的核心。这里我们使用MultinomialNB，通过sklearn库引入：



MultinomialNB是sklearn中多项分布数据的朴素贝叶斯算法的实现，并且是用于文本分类的经典朴素贝叶斯算法。MultinomialNB假设特征的先验概率为[多项式](https://so.csdn.net/so/search?from=pc_blog_highlight&q=%E5%A4%9A%E9%A1%B9%E5%BC%8F" \t "https://blog.csdn.net/TeFuirnever/article/details/_blank)分布，即如下式：



因此，在在MultinomialNB实例化时，有以下三个常用的可选参数:

1. alpha：浮点型可选参数，默认为1.0，其实就是添加拉普拉斯平滑，即为上述公式中的λ ，平滑因子。当小于1时做的是Lidstone平滑；当等于0时，不做任何平滑处理。
2. fit\_prior：布尔型可选参数，默认为True。布尔参数fit\_prior表示是否要考虑先验概率，如果是False，则所有的样本类别输出都有相同的类别先验概率。否则可以自己用第三个参数class\_prior输入先验概率。
3. 可选参数，默认为None。

这里根据我们不断调整参数得到的结果，最终发现参数取下列时有最高的maro-f1和micro-f1值。



MultinomialNB类中的fit函数实现了朴素贝叶斯分类算法训练模型的功能，predict函数实现了法模型预测的功能。

fit函数有如下参数：

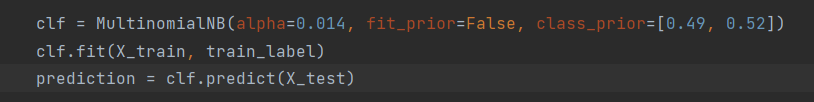
X：大小为[样本数量 , 特征数量]的ndarry，存放训练样本

Y：值为整型，大小为[样本数量]的ndarray，存放训练样本的分类标签

而predict函数有一个向量输入：

X：大小为[样本数量,特征数量]的ndarry，存放预测样本

最后使用fit以及prdict函数预测的代码为：



因此，我们最终采用的贝叶斯预测模型预测函数需要传入三个参数：

1. train\_content:原始训练样本，类型为ndarray；
2. train\_label:训练标签，类型为ndarray；
3. test\_sample：原始测试样本，类型为ndarray。

结果返回一个预测的标签列表prediction。

因为我们得到处理之后的数据集是两个json文件的形式,同时需要将得到的label标签列表数据存入文本当中。因此我们需要一个读取json当中content和label的函数read\_data。他将json当中的content和label分别存入两个列表当中，最后返回两个列表content和label。

最后我们利用f.write函数和for循环将得到的预测label列表结果写入label.txt文本当中。

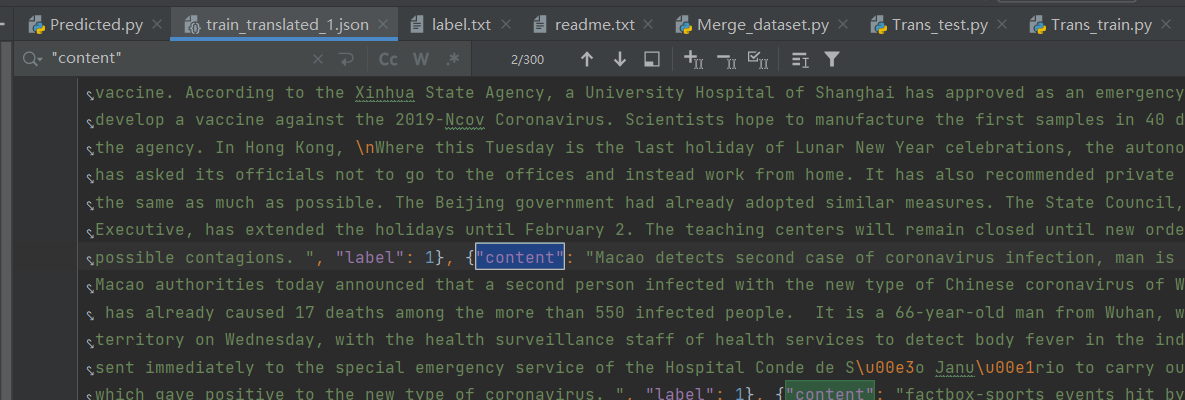
至此对于test测试集的预测分析已经大部分完成，接下来需要做的操作便是将得到的label复制到educoder测试平台，得到micro-f1和marco-f1值，根据得到的值不断调整MultinomialNB中传入的参数值，使预测结果达到相对最佳效果。

## 实验结果

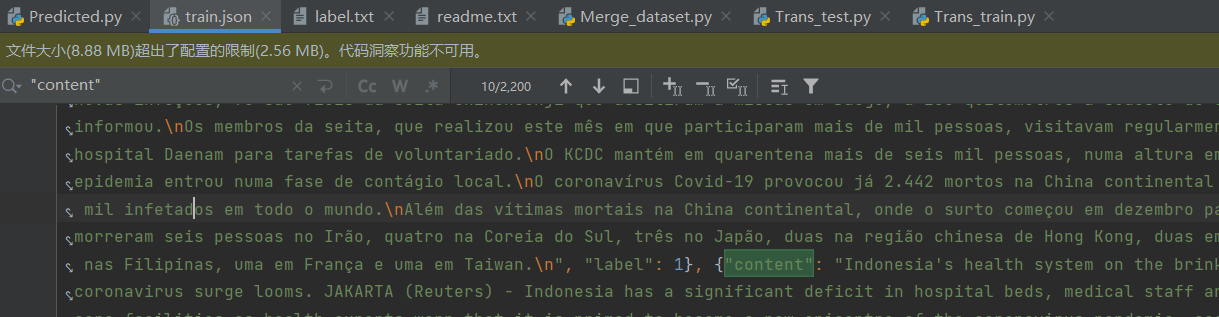
### 6.1数据集翻译结果

调用trans\_train.py和trans\_test.py之后，得到了6个train的json文件和2个test文件。

每个trans\_train.json文件内容形式入下：



对比原始数据集train.json：



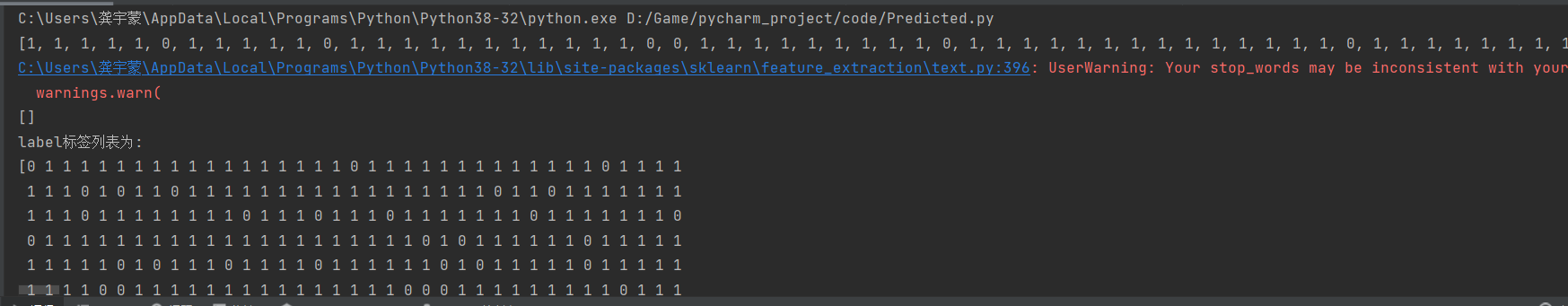
可以很明显的发现，原始数据集各种语言类型的content都被翻译成了英文内容，说明翻译数据是正确执行的。

同样我们查看翻译之后的test.json文件可以得到相似的结果。

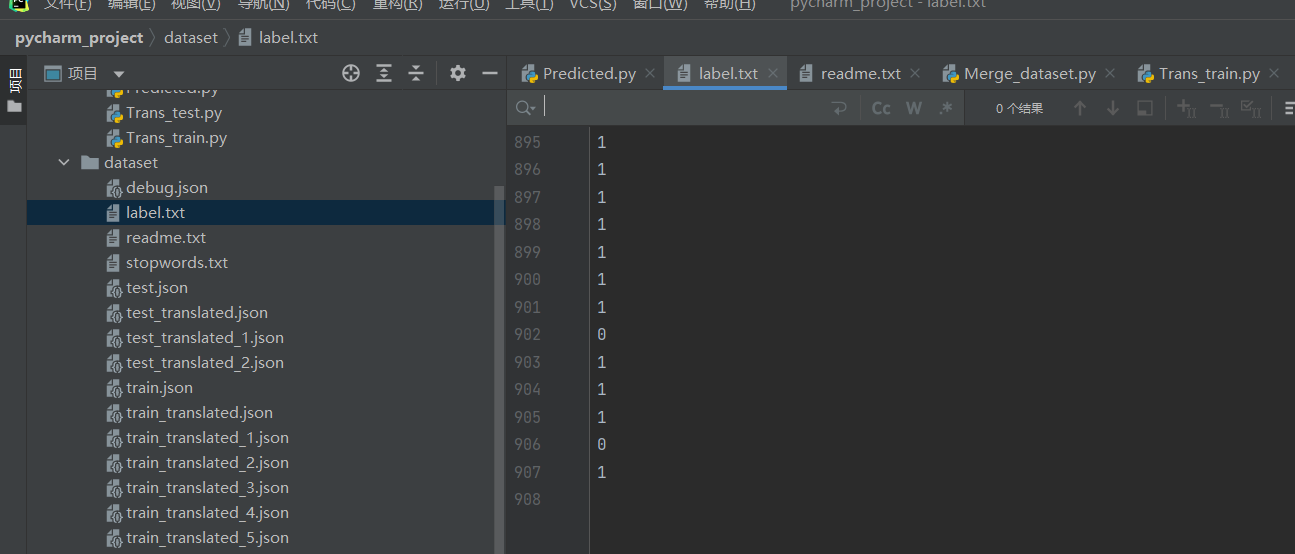
之后我们将得到的分割之后的数据集进行重新连接，利用Merge\_dataset.py得到翻译之后的test\_translated.json和train\_translated.json。

### 6.2贝叶斯分类模型预测结果

在使用了停用词处理、文本向量化、MultinomialNB之后，得到了预测test的label标签列表为：



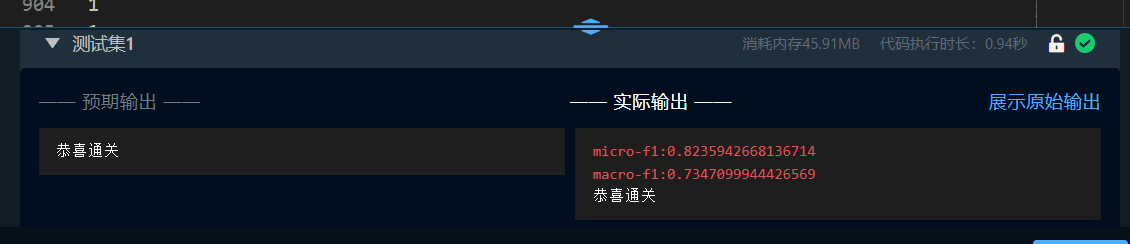
在label.txt文本当中查看可知：



输出的结果共有907个，即对应test数据集中907条数据的label。说明预测程序是正常运行产生label列表的。

### 6.3 educoder测试结果

将得到的label标签结果传入educoder平台测试，得到了如下的预测micro-f1和marco-f1值：



## 结果分析

### 7.1输出结果

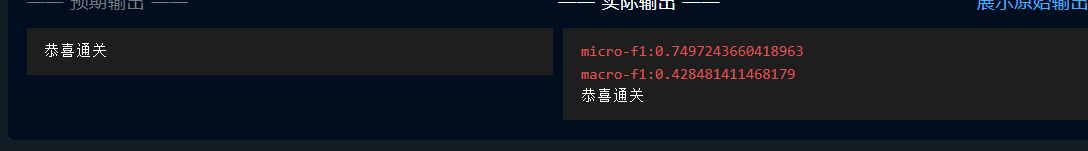
根据我们educoder得到的结果可知我们得到了micro-f1和marco-f1值。但是此值并不是实际的预测准确度，我们如何衡量我们的预测结果呢？

F1-score：是统计学中用来衡量二分类模型精确度的一种指标，用于测量不均衡数据的精度。它同时兼顾了分类模型的精确率和召回率。F1-score可以看作是模型精确率和召回率的一种加权平均，它的最大值是1，最小值是0。

在多分类问题中，如果要计算模型的F1-score，则有两种计算方式，分别为micro-F1和macro-F1，这两种计算方式在二分类中与F1-score的计算方式一样，所以在二分类问题中，计算micro-F1=macro-F1=F1-score，micro-F1和macro-F1都是多分类F1-score的两种计算方式。

由于marco-f1不受数据不平衡影响，容易受到识别性高（高recall、高precision）的类别影响,观察我们的数据可知，train.json的label是极不平衡的。其中的1和0的比例高达10：1，因此这里我们主要将marco-f1的值作为我们实际预测结果的衡量标准。对于我们的算法来说，预测模型精准度越高，marco-f1值越高。

开始时不引入停用词，采用拉普拉斯平滑时(alpha=1)得到的预测marco-f1如下：



可以发现这个值相比于我们最终优化后的值，marco-f1低了很多。因此我们需要对我们的模型进行一定的优化操作，同时改进MultinomialNB函数的平滑因子参数，看能否提高marco-f1的值。

### 7.2算法改进和调参优化

对于朴素贝叶斯算法，由于采用sklearn库中的MultinomialNB，基本过程就是用CountVectorizer将文本转化为词向量，用MultinomialNB实现文本的分类。

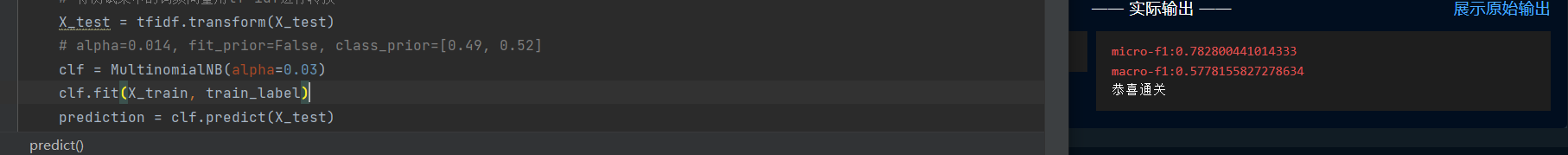
所以我能想到的的优化方法只有放在了对于数据集的进一步处理上。由Googletranslator我们得到了翻译后的数据。但是我们的数据均为新闻文本。考虑到这些文本的特征，文本中会存在大量的重复的人称词和介词等等，这些词在文本向量化时对于预测结果是不必要的，将他们向量化只会影响文本分类处理的结果。因此我们考虑到使用停用词处理数据的content，让content在向量化时去除掉停用词，这样得到的文本向量将更符合我们的预测需要。

根据下载的stopword.txt表可知，一共有1200个左右的停用词，将停用词一行一行的读入stopwords列表，并将stopwords列表作为参数传入CountVectorizer

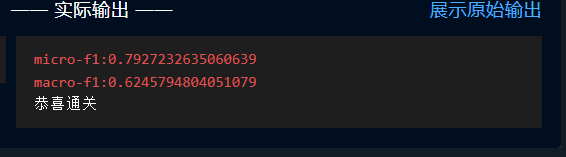
当中。之后的预测分析过程保持不变。

同时可以知道我们使用alpha=1，即单纯的拉普拉斯平滑拟合是不太合适的。因此我们需要不断的调制alpha参数值，选取合适的alpha值让marco-f1尽可能的大。

可以看到alpha=0.03时，marco-f1为0.57

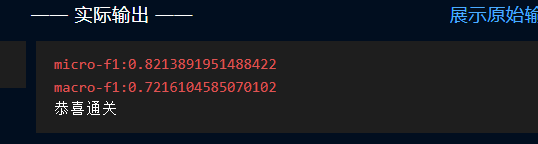


而alpha=0.01时，marco-f1为0.62

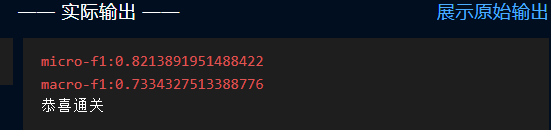


可以很直观的发现，对于alpha的选择会直接影响到最后输出的marco-f1值。通过不停的尝试，我将alpha的范围锁定在了0.01-0.015范围之类，在这个范围得到的marco-f1已经到了可以取到的最大值。

这里我们可以确立了alpha的大致取值范围。但是根据上面我们对于MultinomialNB的介绍，他还有两个参数fit\_prior和class\_prior。我们修改这两个参数能否得到进一步提高呢？于是我尝试将fit\_prior设为False,输入一个先验概率class\_prior=[0.4,0.6].再次预测得到结果的marco-f1如下:

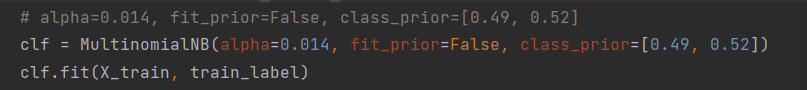


进一步缩小先验概率的范围，这里经过多次尝试将其设置为[0.49,0.52].再次测试得到结果：



这里基本上已经得到了最高的marco-f1值。相比于我们第一次没有引入停用词，alpha设置为1时得到的marco-f1，足足提高了0.3左右。因此对于原始的贝叶斯模型进行一定的优化和参数调整是非常有必要的一件事情。

最终，我们将参数调整为如下值：



得到了最大的marco-f1。是我预想的可以获得的最好的预测结果了。基本上能做的优化处理和参数调整也就到这里了。

## 机器学习的课程体会和建议

### 8.1 结课实验总结

作为机器学习课程的结课大作业，相比于前两次实验内容，这次结课实验更完整。前两次实验都不能算完整的项目内容，只是为了加深课堂学习内容的理解，本次结课大作业是我第一次做完一个机器学习的完整项目，也让我大致了解了相关的项目实现过程。从数据清洗、到选择合适分类模型、再到根据预测结果进行完善改进，每一步的推进都是艰难又充实的。让我影响最深刻的就是对于数据的清洗翻译过程了。之前从来没有使用过翻译api，也是我第一学习了解如何使用翻译API，让我了解了许多相关的知识。之前只是听说python有很多功能强大的库可以调用，这次实验我便亲自感受到了库的强大。不论是json库，还是机器学习模型引入的sklearn库，或是用于处理分割过长content内容的textwrap库，都给我带来了极大的方便。

这次实验也让我学会了观察数据集。虽然这次使用的数据集条数没有那么多，但是我们还是需要对数据集分类进行处理。学会观察处理数据也是非常重要的一件事。

在选择分类的模型时，我多次上网查阅了相关的资料。结合了上课所讲的内容，了解了KNN和SVM以及Bayse等算法模型以及使用方法。最终选择了对于文本分类非常合适的朴素贝叶斯算法。机器学习之所以能不断的发展，也依赖于人们对于各种算法的研究和改进。

总之，第一次接触这种完整的项目，开始过程时非常艰难的。当然做完项目获得的喜悦感也是其他项目不能比拟的。希望通过这次项目所学到的知识能让我不断地受用，在之后的学习中能进一步提升。

### 8.2课程总结

第一次接触到相应的课程，开始学习时充满了新鲜感。当然上课学到的数学基础内容也让人感到困惑。总的来说，机器学习这门课还是带给了我很多的收获的。不仅让我初步学习了相应的算法，完成了第一次接触的机器学习项目，更让我收获了很多经验教训。当然，我认为相对于复杂又需要基础的机器学习课程，40个课时相对有点少。很多算法需要我们了解一下源码的实现过程，但是由于学时的原因，上课没法展开这些内容，只能单纯的讲解理论部分，我认为这时是一个遗憾。希望在之后开设课程时能考虑到这一点，将理论内容与源码讲解结合起来，让学生们更好的理解机器学习课程。

同时作为一个之前没有学过python的学生，学院也没有为我们开设python课程。在开始接触机器学习时对于python语法的适应是非常痛苦的。初步学习了一点python知识后，我才开始做本次结课项目。因此我希望学校能够为下一届学生们安排python选修课，作为机器学习课程学习的基础先导。

这些便是我对于这门机器学习课程学习的一点建议和想法。也希望这门课程能不断改进，越办越好！