**华中科技大学计算机科学与技术学院**

**《机器学习》课堂三结课报告**



专 业： 计算机科学与技术

班 级： 1903

学 号： U201914978

姓 名： 钟逸

成 绩：

指导教师： 何琨

**完成日期： 2021年12月4日**

目录

[新闻数据情感预测 2](#_Toc89788434)

[一、 实验题目：新闻数据情感预测 2](#_Toc89788435)

[二、实验要求 2](#_Toc89788436)

[三、算法设计 2](#_Toc89788437)

[四、实验环境与平台 3](#_Toc89788438)

[五、程序实现 3](#_Toc89788439)

[六、实验结果 3](#_Toc89788440)

[七、结果分析 3](#_Toc89788441)

# 新闻数据情感预测

## 实验题目：新闻数据情感预测

## 二、实验要求

**2.1题目背景**

近年来各种社交平台陆续兴起，网络上的内容越来越多，大量的文本信息涌入互联网，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，进行分析并进一步判断具有一定的潜在价值。因此近年来情感分析受到计算机语言学领域研究者们的密切关注，逐渐成为一项热点研究任务。

本题作为机器学习课程的大作业，旨在对机器学习的知识进行考察。在并不大的数据集中区分文本的情感极性，将新闻分为正负两类。

**2.2数据集**

数据集被分为用于训练的 2200 条新闻（存于train.json中）与用于测试的 907 条新闻（存于test.json中），label 由 0 或 1 表示，其中 0 代表负面（negative），1 代表正面（positive）

**2.3任务描述**

使用机器学习相关知识完成新闻数据情感预测，给定新闻文本，输出相应的情感类别。

**2.4评测标准**

采用F1 score指标，正样本为1，取最后一次提交的分数。

## 三、算法设计

**3.1数据处理**



图3.1.1 文件格式转换



图3.1.2 文本数据清洗

**3.2模型算法**

（1）朴素贝叶斯分类器：

先使用CountVectorizer()将词语转换为词频矩阵，再通过TfidfTransformer()计算词频逆反文档频率。

建立多项式朴素贝叶斯分类器，对拉普拉斯平滑系数alpha作出限定，进行测试。

（2）KNN算法：

使用K近邻算法KNeighborsClassifier进行分类，对n\_neighbors作出限定，进行测试。

（3）支持向量机：

使用LinearSVC进行分类，损失函数选择squared\_hinge，最大迭代数选择10000，选择优化所有类的联合目标，对错误项惩罚参数、多类策略、最大迭代数作出限定，其余参数均为缺省值，进行测试。

**3.3模型评估**

在预测完成后，将结果上传至educoder对结果进行评估，记录每次的分数并以此进行优化。

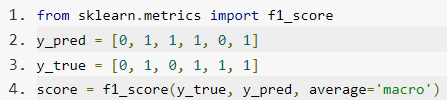


图3.3.1 评估代码参考

## 四、实验环境与平台

系统：Windows10

语言：Python

IDE：PyCharm Community Edition 2021.2.3

Python解释器：Python 3.6

## 五、程序实现

**5.1翻译**

因为数据集中有其他的语言，所以需要先将所有语言转换为英语。

在百度平台中申请翻译API服务，创建translate文件，用于实现翻译translate函数。

首先输入申请服务后得到的用户名以及姓名，再设置转换语言以及终端。



图5.1.1 translate.py初始设置

通过make\_md5函数获取签名。



图5.1.2 translate.py初始设置

编写translate函数，先生成随机数salt，以此获得签名sign，与百度API进行数据交换。



图5.1.3 translate函数实现

**5.2数据处理**

（1）open\_file(file)：将file指示的json文件打开，返回文件对象；

（2）get\_data(file)：用json.load()读取file对象，返回文本数据；

（3）is\_en(content)：使用langdetect工具包中的detect对传入的content文本进行识别，若是英语则返回true，否则返回false；

（4）translate\_to\_en(content)：调用is\_en对文本进行判断，如果是英文，则不做处理，否则调用translate()函数进行翻译。

（5）train\_data\_to\_csv(dataset)：对dataset内容进行循环读取，调用translate\_to\_en()对content进行翻译，结果保存在content中，再和label一同用dataframe存入csv文件，作为处理后的train存放；

（6）test\_data\_to\_csv(dataset)：与train\_data\_to\_csv相同，只是，将结果存放于test.csv文件中；

（7）resolve\_train()：设置文件目录，打开文件，将文件文本数据用np.array()读出存于data中，再调用train\_data\_to\_csv产生train.csv文件；

（8）resolve\_test()：与resolve\_train()相同，只是读取文件目录和输出文件目录不同；

（9）abort\_punctuation(content)：利用str.maketrans()将标点全部替换为逗号；

（10）split\_words(content)：调用nltk.word\_tokenize()对content进行分词处理；

（11）abort\_stop\_words(content)：对content中每个字符串利用stopwords.words()进行判断，根据返回值，选择是否写入；

（12）stemming(content)：调用nltk.stem.SnowballStemmer()，参数传english，生成提取器，对content进行筛选；

（13）clean\_train()：将train.csv文件数据读入frame，再分别将文本内容传给contents，标签传给labels，循环对contents每项内容挨个调用lower()、abort\_punctuation()、split\_words()、abort\_stop\_words()、stemming()，进行文本清洗，将结果和label一起写入resolved\_train.csv文件；

（14）clean\_test()：与clean\_train()无甚区别，只是文件目录的区别；

（15）主函数：依次调用resolve\_train()、resolve\_test()、clean\_train()、clean\_test()，完成训练集和测试集的数据处理。

**5.3训练与预测**

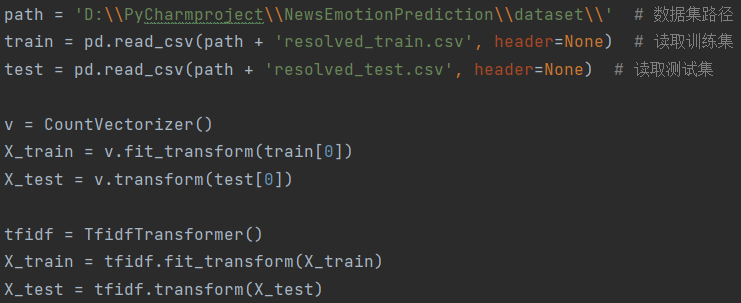


图5.3.1文本数据预处理

（1）贝叶斯：

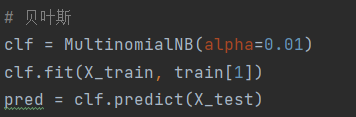


图5.3.2 贝叶斯模型训练及预测

（2）KNN：

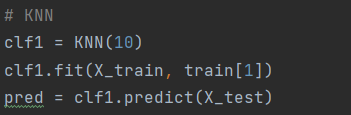


图5.3.3 KNN训练及预测

（3）支持向量机：

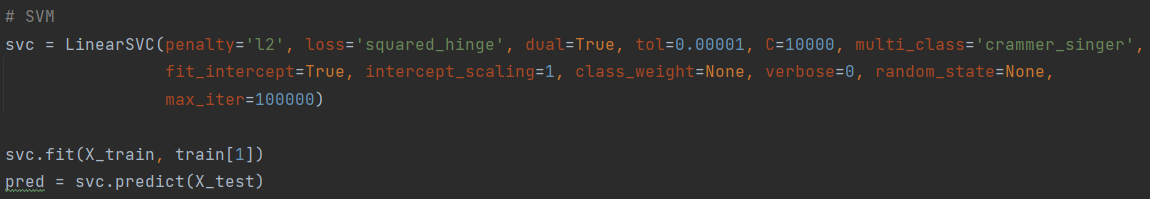


图5.3.4 SVM训练及预测

在完成预测之后，将结果pred写入各自的csv文件中，进行评估。

## 六、实验结果

**6.1数据处理**

（1）翻译训练集：

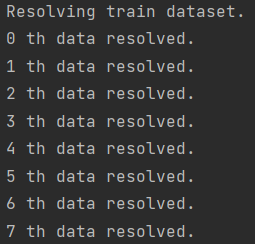


图6.1.1 训练集处理开始



图6.1.2 训练集处理结束

（2）翻译测试集：

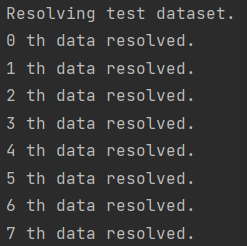


图6.1.3 测试集处理开始

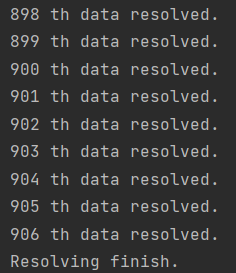


图6.1.4 测试集处理结束

（3）清洗训练集数据：

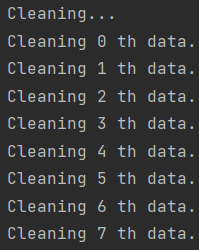


图6.1.5 训练集清洗开始

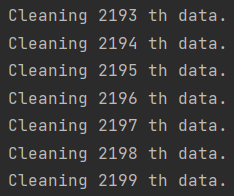


图6.1.6 训练集清洗结束

（4）清洗测试集数据：

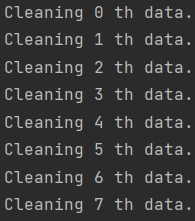


图6.1.7 测试集清洗开始

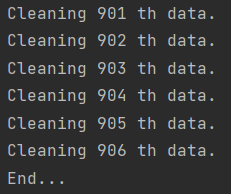


图6.1.8 测试集清洗结束

**6.2预测模型**

（1）贝叶斯：**MultinomialNB()**

表6.2.1 贝叶斯评估

|  |  |
| --- | --- |
| alpha | F1 score |
| 1.0(default) | 0.428481 |
| 0.1 | 0.442336 |
| 0.01 | 0.635537 |
| 0.001 | 0.630545 |

（2）KNN：**KNeighborsClassifier()**

表6.2.2 KNN评估

|  |  |
| --- | --- |
| n\_neighbors | F1 score |
| 3 | 0.630128 |
| 4 | 0.661173 |
| 5(default) | 0.589111 |
| 6 | 0.632117 |
| 7 | 0.575046 |
| 8 | 0.589111 |
| 9 | 0.569221 |

（3）SVM：**LinearSVC()**

对max\_iter进行考察：

表6.2.3 SVM评估：max\_iter

|  |  |  |  |
| --- | --- | --- | --- |
| max\_iter | C | multi\_class | F1 score |
| 1000(default) | 1(default) | Crammer\_singer | 0.661509 |
| 10000 | 1 | Crammer\_singer | 0.661509 |
| 100000 | 1 | Crammer\_singer | 0.661509 |
| 1000 | 1 | ovr | 0.639520 |
| 10000 | 1 | ovr | 0.639520 |
| 100000 | 1 | ovr | 0.639520 |

对C和multi\_class进行考察：

表6.2.4 SVM评估：C

|  |  |  |  |
| --- | --- | --- | --- |
| C | max\_iter | multi\_class | F1 score |
| 1 | 10000 | Crammer\_singer | 0.661509 |
| 10 | 10000 | Crammer\_singer | 0.672487 |
| 100 | 10000 | Crammer\_singer | 0.653262 |
| 1000 | 10000 | Crammer\_singer | 0.660813 |
| 10000 | 10000 | Crammer\_singer | 0.660813 |
| 1 | 10000 | ovr | 0.639520 |
| 10 | 10000 | ovr | 0.670921 |
| 100 | 10000 | ovr | 0.655963 |
| 1000 | 10000 | ovr | 0.654919 |
| 10000 | 10000 | ovr | 0.658170 |

## 七、结果分析

（1）贝叶斯：按理说我们的样本数量较少，可能会出现测试集某个词在训练集中未出现过的情况，所以考虑引入拉普拉斯平滑系数。alpha默认取1，对拉普拉斯平滑的具体计算较为复杂，所以也依次取0.1、0.01、0.001进行测试，最后发现取0.01时效果最佳。

（2）KNN：取n\_neighbors的值进行测试，因为缺省值为5，所以在5的周围取数，最终发现n\_neighbors=4时效果最佳。

（3）SVM：linearSVC的参数众多，选择了错误项的惩罚参数C、多类策略、最大迭代次数进行测试。

迭代次数缺省值为1000，依次取1000、10000、100000进行评测，可能是数据过少，发现最大迭代数的调整对于结果并没有影响；

惩罚参数C是对结果影响较大的参数，它对模型进行约束，可见，对于crammer\_singer和ovr两个策略，C都在等于10时取得最佳的效果；

多类策略只有两个可选值，在其余条件相同的情况下，crammer\_singer和ovr各有更佳效果的时候，但是最终在所有取值中发现最佳效果是当策略选择crammer\_singer时得到的。crammer\_singer在实践中其实并不经常使用，他的准确性提升并不高，而且计算成本是显著高于ovr策略的，可以感受到它运行的时间更久。

最终综合三种不同分类器观察，SVM是效果最佳的，但其实也并没有显著地提升效果，可能是源于较小的训练集。