机器学习导论大作业报告

--互联网新闻情感分析

组员1（组长） 王晨曦 学号

组员2 蔡润泽 学号

组员3 李昊宸 学号2017K8009929044

组员4 裴梓良 学号

组员5 付轶凡 学号

1. 任务描述
2. 赛题背景

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，情感分为正中负三类。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。

1. 任务要求

对官方提供的新闻数据进行情感极性分类，其中正面情绪对应0，中性情绪对应1以及

负面情绪对应2。根据提供的训练数据，通过算法或模型判断出测试集中新闻的情感极

性。

1. 数据描述

数据包由两个csv文件组成：第一个是Train\_Dataset，包含7360条新闻的id号，新闻标题和新闻内容。第二个是Train\_Dataset\_Label，包含了Dataset中新闻的id号，以其新闻的情感得分（用0，1，2表示）。

1. 实施方案

该问题实质上为对信息的分类处理，所以核心内容是使用一个合适的分类器。其次，由

于新闻是由文本构成的语言，一条新闻的情感通常可以由文本中词语的情感性决定。于

是，另一个重要的内容是如何将数据进行预处理，即删除无用文字，并将新闻文本切分

成一个个中文词语。

1. 数据预处理

观察训练集中新闻的内容，发现新闻文本乱七八糟，有各种不属于中文词库的符号。所以预处理的第一步就是将不属于中文的文本删除（包括各种标点符号）。预处理的第二步是将修正后的文本进行词语的切分，从而将一整段话切分为一个个词语。

1. 分类器选择

情感标签有三种赋值：积极、中立和消极。于是所有的二分类器就不可以使用，比如标准意义下的SVM支持向量机等。考虑到运行时间和效率，我们将选择朴素贝叶斯分类器作为首选（事实上，测试结果也表明朴素贝叶斯分类器是效率和正确率均较高的分类器）

1. 具体实现

实现细节，模型设计和选择，数据预处理方式

以下是整个处理过程的具体实现：

1. 数据清理

对非中文无用数据的清理，需要将以下类别的数据从训练集和测试集中清除：

html  = re.compile('<.\*?>')

    http  = re.compile(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-\_@.&+]|[!\*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')

    src   = re.compile(r'\b(?!src|href)\w+=[\'\"].\*?[\'\"](?=[\s\>])')

    space = re.compile(r'[\\][n]')

    ids   = re.compile('[(]["微信"]?id:(.\*?)[)]')

    wopen = re.compile('window.open[(](.\*?)[)]')

    english = re.compile('[a-zA-Z]')

    others= re.compile(u'[^\u4e00-\u9fa5\u0041-\u005A\u0061-\u007A\u0030-\u0039\u3002\uFF1F\uFF01\uFF0C\u3001\uFF1B\uFF1A\u300C\u300D\u300E\u300F\u2018\u2019\u201C\u201D\uFF08\uFF09\u3014\u3015\u3010\u3011\u2014\u2026\u2013\uFF0E\u300A\u300B\u3008\u3009\!\@\#\$\%\^\&\\*\(\)\-\=\[\]\{\}\\\|\;\'\:\"\,\.\/\<\>\?\/\\*\+\\_"\u0020]+')

需要将html链接、数据来源、用户名、英语字符和其他单个非中文字符清除，采用以

上正则表达式描述需要删除的类别，配合sub命令将其从训练集和数据集中删除。

1. 数据划分

数据划分使用的库为jieba分词，具体的操作如下：

if \_\_name\_\_ =='\_\_main\_\_':

    jieba.load\_userdict('dict.txt')

    jieba.enable\_parallel(2)

    print("Processing: cutting train data...")

    cut\_Train\_Data = cutData('Train/preprocessed\_train\_data.csv')

    cut\_Train\_Data.to\_csv('Train/preprocessed\_train\_data.csv')

    print("Processing: cutting test data...")

    cut\_Test\_Data = cutData('Test/Test\_DataSet\_P.csv')

    cut\_Test\_Data.to\_csv('Test/result.csv')

我们使用的词典并非jieba的内置词典，而是用户自定义词典dict.txt，分词过程开双线程优化。

分词函数cutData的定义：

def cutData(filePath):

    cutData = pd.read\_csv(filePath,index\_col=0)

    cutData['title'] = pd.DataFrame(cutData['title'].astype(str))

    cutData['title'] = cutData['title'].apply(lambda x: cut\_char(x))

    cutData['content'] = pd.DataFrame(cutData['content'].astype(str))

    cutData['content'] = cutData['content'].apply(lambda x: cut\_char(x))

    cutData['combine'] = cutData['content']+'/'+70\*(cutData['title']+'/')

    print(cutData.head())

    return cutData

分别对数据集的title和content进行分词，并在处理结束之后将二者重新拼接，中间用“/”隔开。在后序的实验中我们发现，标题比文章更能准确地表述新闻的情感（在不碰到UC浏览器的新闻的情况下），所以使用70次重复标题中词语的内容来加大重要性（即其出现频率）。分词方式使用jieba的全精度分词方式。

1. 构建词频矩阵

经过以上两步，我们已经将文本成功分割为独立的中文词语，接下来需要统计每个词出现的频率及分布。

stop\_words = get\_custom\_stopwords('ChineseStopWords.txt')

首先需要获得停用词表。这里我们使用的是百度停用词表、哈工大停用词表、中文停用词表等多个词表的综合结果。

Vectorizer = CountVectorizer( max\_df = 0.8,

                                  min\_df = 2,

                                  token\_pattern = u'(?u)\\b[^\\d\\W]\\w+\\b',

                                  stop\_words =frozenset(stop\_words)

                                  )

    Vectorizer\_Title = CountVectorizer( max\_df = 0.8,

                                  min\_df = 3,

                                  token\_pattern = u'(?u)\\b[^\\d\\W]\\w+\\b',

                                  stop\_words =frozenset(stop\_words) )

CountVectorizer是通过fit\_transform函数将文本中的词语转换为词频矩阵，矩阵元素a[i][j] 表示j词在第i个文本下的词频。即各个词语出现的次数，通过get\_feature\_names()可看到所有文本的关键字，通过toarray()可看到词频矩阵的结果。

X = trainData['content'].astype('U')

    y = trainData.label

    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=2019)

test = pd.DataFrame(Vectorizer.fit\_transform(X\_train).toarray(), columns=Vectorizer.get\_feature\_names())

先对文本内容进行词频统计。值得一提的是，在后序的操作中我们相继进行了对标题的词频统计和综合标题和文本的词频统计，并针对这三个矩阵的统计结果进行了三次贝叶斯分类。

此处需要说明的是，我们将训练集中随机选出的80%数据用于分类器的训练，剩余20%用于测试训练的结果。

1. 朴素贝叶斯分类

朴素贝叶斯方法是基于贝叶斯定理的一组有监督学习算法，即“简单”地假设每对特征之间相互独立。 给定一个类别 IMG_256 和一个从 IMG_257 到 IMG_258 的相关的特征向量， 贝叶斯定理阐述了以下关系:

IMG_259

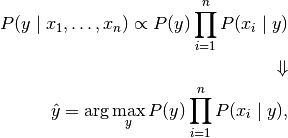
使用简单(naive)的假设-每对特征之间都相互独立:

IMG_260

对于所有的 i 都成立，这个关系式可以简化为

IMG_261

由于在给定的输入中 IMG_262 是一个常量，我们使用下面的分类规则:

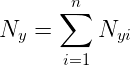


我们可以使用最大后验概率(Maximum A Posteriori, MAP) 来估计 IMG_264 和 IMG_265 ; 前者是训练集中类别 IMG_266的相对频率。

回到原问题，考虑到我们分类的目标不是某一个词语，而是基于文本生成的词向量，于是在选择具体的分类器时我们选择了多项分布朴素贝叶斯。多项分布朴素贝叶斯的特征是分布参数由每类 IMG_256的 IMG_257 向量决定， 式中 IMG_258 是特征的数量(对于文本分类，是词汇量的大小)， IMG_259 是样本中属于类 IMG_260 中特征 IMG_261 概率 IMG_262 。

参数 IMG_263 使用平滑过的最大似然估计法来估计，即相对频率计数:

IMG_264

式中IMG_265是 训练集T中特征i在类IMG_266中出现的次数， 是类 IMG_268 中出现所有特征的计数总和。这样一来，我们可以通过一条新闻内出现各种词的频率和后验概率对该新闻进行情感分类。

nb = MultinomialNB()

    X\_train\_vect = Vectorizer.fit\_transform(X\_train)

    nb.fit(X\_train\_vect, y\_train)

    train\_score = nb.score(X\_train\_vect, y\_train)

    print("content train score is : ",train\_score)

接下来进行多项分布朴素贝叶斯的训练。

Content train score反映的是分类器在训练数据上的得分情况（以分类正确的比例为参考）

1. 将训练数据应用到测试集上

X\_test\_vect = Vectorizer.transform(X\_test)

    print("content test score is : ", nb.score(X\_test\_vect, y\_test))

    y\_predict = nb.predict(Vectorizer.transform(X\_test))

    print("content test macro f1\_score:",sklearn.metrics.f1\_score(y\_test, y\_predict, average='macro'))

Content test score反映的是分类器在其他随机选出的20%训练集数据上的得分情况（以分类正确的比例为参考）

给出训练器在训练集中的20%测试集上的F1-score。F1-score : 2(PR)/(P+R）

其中各符号的含义：

准确率(P) ： TP/ (TP+FP)

召回率(R) ： TP(TP + FN)

真阳性（TP）: 预测为正， 实际也为正

假阳性（FP）: 预测为正， 实际为负

假阴性（FN）: 预测为负，实际为正

真阴性（TN）: 预测为负， 实际也为负

    from sklearn.metrics import confusion\_matrix

    cm = confusion\_matrix(y\_test, y\_predict)

    print(cm)

打印分类器在训练集中的20%测试集上分类的混淆矩阵。

1. 对目标集进行分类

print("Apply to Test Data...")

    testData = pd.read\_csv('Test/result.csv',index\_col=0)

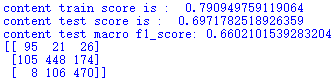
    testResult = nb.predict(Vectorizer.transform(testData['content'].astype('U')))

    testData['label\_content'] = testResult

1. 实验结果及分析
2. 数据集划分方式

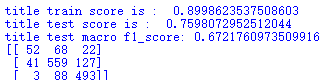
数据集的划分采用朴素的随机采样方式，选取随机种子，将样本训练集的80%用来训练，剩余20%用来测试训练后分类器的性能。

1. 处理结果
2. 对新闻文本进行训练的结果



可以看出，仅对文本进行训练的效果不是很好，对训练集自身的正确分类率有79%，对保留出来的测试集的正确分类率也只有70%，F1得分为0.66。从混淆矩阵中可以看出，分类器对中立态度的分类情况不甚理想。

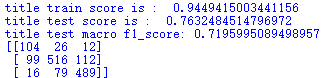
1. 对新闻标题进行训练的结果



仅对新闻标题进行训练，性能有较大提升。对训练集自身的正确分类率提高11个百分点，对保留测试集的提升也有6%，F1得分有0.012的提高，变动不大。分析混淆矩阵，发现新闻标题的分类在中性和负面情绪的分类性能有较大提升，但对正面情绪的分类效果较差（甚至和随机分类差不多）。

1. 对新闻文本和新闻标题联合训练的结果

由上面两次训练的结果可知，文本分类的缺点在中性情绪，标题分类的缺点在正面情绪，所以按权重将文本和标题联合训练应该会将二者的问题避免。考虑到新闻标题的长度与新闻文本的长度相去甚远，我们采用了在数据预处理时将新闻标题多次重复出现的方式适当提高新闻标题中词语的词频（我们使用的是重复70次）。



经过修正后，对训练集自身的正确分类率达到94.5%，对保留测试集的分类达到76.3%，F1得分为0.72，达到了较为理想，平均分类水平差距较小的结果。

1. 总结展望

总的来说，基于朴素贝叶斯分类的词袋模型分类器分类效果还算较为理想，在计算时间

短的优点上可以达到较高的分类精度。相比我们使用神经网络分类器和聚类分类器的尝

试，首先分类器的处理结果并不直观，其次训练时间非常长，最终得到的结果也与贝叶

斯分类器的结果不尽相同。

在处理数据的过程中，我们也发现了一些导致分类结果难以突破75%大关的门槛：词语

的多意性，甚至是情感的完全相反性。比如说，在某新闻标题和文本中提到了“闹新春”

的内容，分类器将“闹”一字识别为负面情绪，并且带有很高的向量权重，导致该条新

闻原本想体现的是正面情绪，却直接被分到了负面情绪之中。此外，当正负面词语同时

出现时，是将其分到正面、负面还是中立对于分类器来说还是很难实现的。

总而言之，基于word2Vec思路下的词语情感划分能实现的分类精度大致不会有更大的

提高，查阅相关资料表明这与word2Vec不基于上下文的词义分类有关。如果使用Bert

进行词语的训练或许会有更大的突破。

1. 任务分工

王晨曦：

蔡润泽：python代码编写、调试

李昊宸：模型调试、分析及性能评测、报告撰写

裴梓良：

付轶凡：