# 华东理工大学

# 模式识别大作业

| | |
|---|---|
| 题目 | 电影评论识别 |
| 院系 | 信息科学与工程学院 |
| 专业 | 控制科学与工程 |
| 组员 | 常硕 |
| 指导老师 | 赵海涛 |

# 电影评论识别

## 一、电影评论识别简介

**题目描述：**

对电影的评论文字的处理是自然语言处理中情感分析方向的一个很好的例子。人们的语言经常具有不确定性，有的甚至包括了讽刺、反语等等。如何识别出评论者的态度是一大难点。

在这道题中，给出电影评论，要求识别出这条评论是积极的还是消极的。对于文字处理的方法一般为：第一步从自然语言中提取出特征，第二步对特征进行适当的分类器训练。（在分类器的选择中建议采用神经网络等比较强的分类器。）

第一步特征提取可以尝试采用 Google 的 Word2Vec 方法，Word2Vec 试图理解单词之间的含义和语义关系。它的原理类似于递归神经网络，但计算效率更高。此外也可以使用 TF-IDF、词袋法等等。

## 二、题目评价和解决方案

本题是一道自然语言处理当中的情感分析。可以先从句子当中提取出特征，方法如 Word2Vec，TF-IDF，词袋法等等。提取出特征后，可以采用朴素贝叶斯，朴素贝叶斯在文本处理当中有比较好的表现效果。也可以采用 SVM、递归神经网络等较强的分类器。通过该分类器，给出一个概率值，这个值不需要再进一步处理为 0-1 二值，因为这里采用 AUC 进行评价。

想要很好的完成这个任务：要掌握词袋法、TF-IDF 或 Word2Vec 等文本特征提取技术。掌握循递归神经网络、朴素贝叶斯或其他分类器。理解 AUC 评价指标。会 svm 或者 rnn 等分类器的使用等。

在这个题目中我尝试用词袋模型和 TF_IDF 模型进行文本进行特征工程处理。用逻辑回归法和多项式朴素贝叶斯分类器对模型处理后的文本数据进行机器学习和数据挖掘。最后得到在词袋模型下用 sklearn 默认的逻辑分类器和 sklearn 默认的多项式朴素贝叶斯分类器分别得到的预测准确率为 0.89312 和 0.88288。在 TF_IDF 模型得到的文本特征在逻辑回归分类器和朴素贝叶斯分类器下的预测概率分别为 0.8896 和 0.90544。注意这里逻辑回归是经过网络搜索

后找到了最优参数组合对应的逻辑回归模型。训练好之后发现逻辑回归效果好一些就采用逻辑回归得到测试集的结果。

其实，这道题目相对比较简单。要求将每条电影评论抽象为一个 0-1 分类问题，就是把这个问题看成一个分类问题，利用数据，找出影响影评的消极或者积极的因素，并且对这些“因素”与“积极评论或消极评论”这两者关系进行建模。

通过建立模型后，在训练数据上反复调试模型中的部分参数，使得分类器对于训练数据达到较好的分类效果后，将该模型用于测试数据的预测，并分析其预测效果。

## 2.1 数据导入

数据从 lintcode 网站获取，我用的 labeledTrainData.tsv 进行训练，testData.tsv 进行测试。

| | id | sentiment | review |
|---|---|---|---|
| 0 | 5814_8 | 1 | With all this stuff going down at the moment w... |
| 1 | 2381_9 | 1 | \The Classic War of the Worlds\" by Timothy Hi... |
| 2 | 7759_3 | 0 | The film starts with a manager (Nicholas Bell)... |
| 3 | 3630_4 | 0 | It must be assumed that those who praised this... |
| 4 | 9495_8 | 1 | Superbly trashy and wondrously unpretentious 8... |

**图 1 部分训练样本信息**

| | id | review |
|---|---|---|
| 0 | 3862_4 | I just watched it. A couple of laughs, but not... |
| 1 | 674_10 | While to most people watching the movie, this ... |
| 2 | 8828_10 | I was so glad I came across this short film. I... |
| 3 | 2963_8 | The creators of south park in their own film h... |
| 4 | 2483_1 | Unspeakably discombobulated turkey, a mix of a... |

**图 2 部分测试样本信息**

其中共有 25000 行训练数据，每行数据都有句子 ID、文本内容、情感标签三列，共有 5000 行测试集数据，每个数据都有句子 ID、文本内容两列。

```
data_train = pd.read_csv('D:\\CHENGXU\\lintcode\\dianying\\labeledTrainData.tsv',sep='\t')
data_test = pd.read_csv('D:\\CHENGXU\\lintcode\\dianying\\testData.tsv',sep='\t')
```

## 2.2 数据处理

构建语料库，需要对文本进行一些处理，将原始文本中的每一个词变成计算机看得懂的向量，这一过程叫做文本的特征工程，非常重要。 有很多将词变成向量的方法，比如下面将要用的词袋模型、TF-IDF 模型，另外 word2vec 模型时间原因我没进行尝试。不管采用什么模型，我们都需要先把训练集和测试集中所有文本内容组合在一起，构建一个语料库。

1、先把测试集和训练集的影评都提取出来在整合到一起。

```
# 提取训练集中的文本内容
train_sentences = data_train['review']

# 提取测试集中的文本内容
test_sentences = data_test['review']

# 通过pandas的concat函数将训练集和测试集的文本内容合并到一起
sentences = pd.concat([train_sentences,test_sentences])
```

一共有 30000 条评论。

2、提取训练集中的‘sentiment’即情感标签一共是 25000 个标签，用于进行文本特征工程。

### 2.2.1 词袋模型

使用词袋模型进行文本特征工程，用 sklearn 库中的 CountVectorizer 构建词袋模型， ngram 指分析相邻的几个词，避免原始的词袋模型中词序丢失的问题，max_features 指最终的词袋矩阵里面包含语料库中出现次数最多的多少个词。

```
from sklearn.feature_extraction.text import CountVectorizer
co = CountVectorizer(
    analyzer='word',
    ngram_range=(1,4),
    max_features=150000
)
```

**图 3 词袋模型**

构建分类器算法，对词袋模型处理后的文本进行机器学习和数据挖掘逻辑回归分类器。

```
from sklearn.linear_model import LogisticRegression
lg1 = LogisticRegression()
lg1.fit(x_train,y_train)
print('词袋方法进行文本特征工程，使用sklearn默认的逻辑回归分类器，验证集上的预测准确率:',lg1.score(
x_test,y_test))
```

```
词袋方法进行文本特征工程，使用sklearn默认的逻辑回归分类器，验证集上的预测准确率: 0.89312
```

构建分类器算法，对词袋模型处理后的文本进行机器学习和数据挖掘多项式朴素贝叶斯分类器。

```
#引用朴素贝叶斯进行分类训练和预测
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(x_train,y_train)
print('词袋方法进行文本特征工程，使用sklearn默认的多项式朴素贝叶斯分类器，验证集上的预测准确率:',cl
assifier.score(x_test,y_test))
```

```
词袋方法进行文本特征工程，使用sklearn默认的多项式朴素贝叶斯分类器，验证集上的预测准确率: 0.88288
```

从上面两个图进行比较：多项式朴素贝叶斯分类器，训练速度很快，但准确率较低；逻辑回归训练速度慢，准确率高。

### 2.2.2 TF-IDF 模型

使用 TF-IDF 模型进行文本特征工程。TF 值衡量了一个词出现的次数。 IDF 值衡量了这个词是不是有用。如果是 the、an、a 等烂大街的词，IDF 值就会很低。两个值的乘积 TF_IDF 反映了一个词的出现带来的特异性信息。

```
from sklearn.feature_extraction.text import TfidfVectorizer
tf = TfidfVectorizer(
    analyzer='word',
    ngram_range=(1,4),
    max_features=150000
)
```

图 4 TF-IDF 模型

构建朴素贝叶斯分类器算法，对 TF-IDF 模型处理后的文本进行机器学习和数据挖掘。

```
#引用朴素贝叶斯进行分类训练和预测
classifier = MultinomialNB()
classifier.fit(x_train,y_train)
print('TF-IDF方法进行文本特征工程，使用sklearn默认的多项式朴素贝叶斯分类器，验证集上的预测准确率:',
classifier.score(x_test,y_test))
```

```
TF-IDF方法进行文本特征工程，使用sklearn默认的多项式朴素贝叶斯分类器，验证集上的预测准确率: 0.8896
```

构建逻辑回归分类器算法，对 TF-IDF 模型处理后的文本进行机器学习和数据挖掘 。

```
# sklearn默认的逻辑回归模型
lg1 = LogisticRegression()
lg1.fit(x_train,y_train)
print('TF-IDF方法进行文本特征工程，使用sklearn默认的逻辑回归模型，验证集上的预测准确率:',lg1.score(
x_test,y_test))
```

```
TF-IDF方法进行文本特征工程，使用sklearn默认的逻辑回归模型，验证集上的预测准确率: 0.89184
```

上图采用的是 sklearn 默认的逻辑回归模型，验证集上的预测准确率较低。

```
# C：正则化系数，C越小，正则化效果越强
# dual：求解原问题的对偶问题
lg2 = LogisticRegression(C=3, dual=True)
lg2.fit(x_train,y_train)
print('TF-IDF方法进行文本特征工程，使用增加了两个参数的逻辑回归模型，验证集上的预测准确率:',lg2.sco
re(x_test,y_test))
```

```
TF-IDF方法进行文本特征工程，使用增加了两个参数的逻辑回归模型，验证集上的预测准确率: 0.90112
```

对比两个预测准确率可以看出，在逻辑回归中增加 C 和 dual 这两个参数可以提

高验证集上的预测准确率，但如果每次都手动修改就太麻烦了。我们可以用 sklearn 提供的强大的网格搜索功能进行超参数的批量试验。 搜索空间：C 从 1 到 9。对每一个 C，都分别尝试 dual 为 True 和 False 的两种参数。 最后从所有参数中挑出能够使模型在验证集上预测准确率最高的。

```
from sklearn.model_selection import GridSearchCV
param_grid = {'C':range(1,10),
              'dual':[True,False]
              }
lgGS = LogisticRegression()
grid = GridSearchCV(lgGS, param_grid=param_grid,cv=3,n_jobs=-1)
grid.fit(x_train,y_train)
```

通过上述计算得到最终 C 为 9，dual 为 Ture 时，找到最优超参数组合对应的逻辑回归模型，此时在验证集上的预测准确率: 0.90544。对比前面的 0.891 和 0.901 在精确度上还是略有提升的。

### 2.3 处理结果

在对测试集进行处理时使用 TF-IDF 对测试集中的文本进行特征工程，之后再对测试集中的文本，使用 lg_final 逻辑回归分类器进行预测。得到一个维度为 5000 的数据组。

```
# 使用TF-IDF对测试集中的文本进行特征工程
test_X = tf.transform(data_test['review'])
```

```
# 对测试集中的文本，使用lg_final逻辑回归分类器进行预测
predictions = lg_final.predict(test_X)
```

| | id | review | sentiment |
|---|---|---|---|
| 0 | 3862_4 | I just watched it. A couple of laughs, but not... | 0 |
| 1 | 674_10 | While to most people watching the movie, this ... | 1 |
| 2 | 8828_10 | I was so glad I came across this short film. I... | 1 |
| 3 | 2963_8 | The creators of south park in their own film h... | 1 |
| 4 | 2483_1 | Unspeakably discombobulated turkey, a mix of a... | 0 |

**图 5 部分测试数据**

最终结果在 final_data 中储存。

### 2.4 实验方法及结论

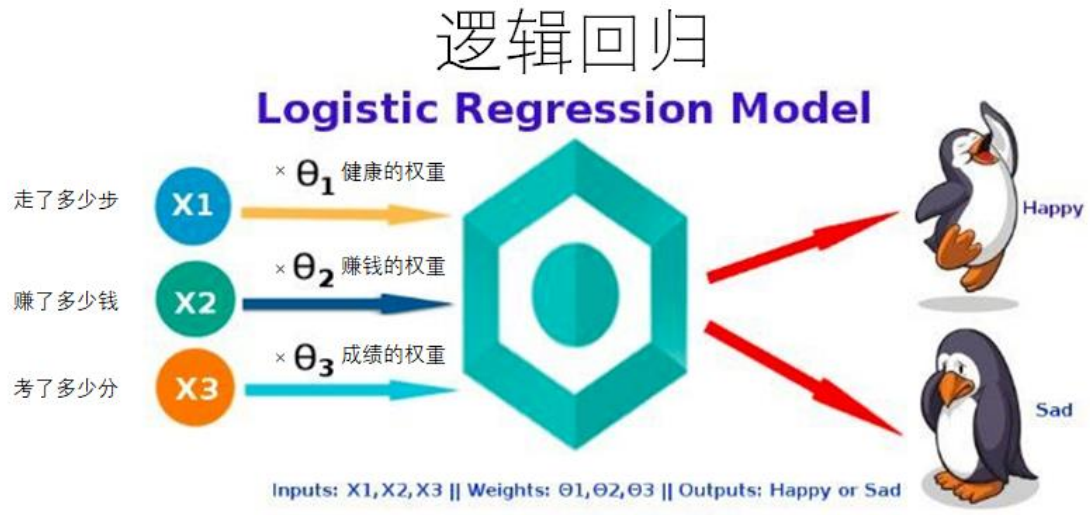
在本次实验中采用了词袋模型和 TF-IDF 模型进行文本特征工程处理，后又

分别用多项式朴素贝叶斯分类器和逻辑回归分类器进行验证集上的准确率预测。从上图和数据中我们得出了：多项式朴素贝叶斯分类器，训练速度很快，但准确率较低；逻辑回归训练速度慢，准确率高。

**（1）朴素贝叶斯**

朴素贝叶斯：朴素贝叶斯法是基于贝叶斯定理与特征条件独立假设的分类方法。最为广泛的两种分类模型是决策树模型(Decision Tree Model)和朴素贝叶斯模型（Naive Bayesian Model，NBM）。和决策树模型相比，朴素贝叶斯分类器(Naive Bayes Classifier,或 NBC)发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。同时，NBC 模型所需估计的参数很少，对缺失数据不太敏感，算法也比较简单。理论上，NBC 模型与其他分类方法相比具有最小的误差率。但是实际上并非总是如此，这是因为 NBC 模型假设属性之间相互独立，这个假设在实际应用中往往是不成立的，这给 NBC 模型的正确分类带来了一定影响。

**（2）逻辑回归**




从上图可以看出逻辑回归简单说就是多个输入，再乘上各自的权重后，经过激活函数进行判断可以得到不同的结果。本实验就是简单的积极和消极分类即 0 和 1 分布。

**（3）TF-IDF 模型**

TF-IDF：TF-IDF 是一种统计方法，用以评估一字词对于一个文件集或一个语料库中的其中一份文件的重要程度。字词的重要性随着它在文件中出现的次数成正比增加，但同时会随着它在语料库中出现的频率成反比下降。TF-IDF 加

权的各种形式常被搜索引擎应用，作为文件与用户查询之间相关程度的度量或评级。除了 TF-IDF 以外，因特网上的搜索引擎还会使用基于链接分析的评级方法，以确定文件在搜寻结果中出现的顺序。

**（3）词袋模型**

- John likes to watch movies.
- Mary likes movies too.
- John also likes football.

| John | likes | to | watch | movies | Mary | too | also | football |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

图中数字分别代表了对应的词在一个评论中出现的次数。

## 三、实验结论

多项式朴素贝叶斯分类器，训练速度很快，但准确率较低；逻辑回归训练速度慢，准确率高。经过多次试验，采用了 TF-IDF 模型进行文本特征工程处理，后用最优参数逻辑回归分类器进行验证集上的准确率预测最高效果最理想。

## 附：文件说明

本次上传内容一共包含有：

1、大作业报告；

2、最终的 python 程序源码：dianying_qinggan.ipynb；

3、训练数据：labeleledTrainData.csv；

4、测试数据;testData.csv；

5、最终预测数据：final_data.csv