程序报告

学号：2211044　　姓名：陆皓喆

一、问题重述

1.1 实验背景

垃圾短信（$SpamMessages，SM$）是指未经过用户同意向用户发送不愿接收的商业广告或者不符合法律规范的短信。

随着手机的普及，垃圾短信在日常生活日益泛滥，已经严重的影响到了人们的正常生活娱乐，乃至社会的稳定。

据 360 公司 2020 年第一季度有关手机安全的报告提到，360 手机卫士在第一季度共拦截各类垃圾短信约 34.4 亿条，平均每日拦截垃圾短信约 3784.7 万条。

大数据时代的到来使得大量个人信息数据得以沉淀和积累，但是庞大的数据量缺乏有效的整理规范；

在面对量级如此巨大的短信数据时，为了保证更良好的用户体验，如何从数据中挖掘出更多有意义的信息为人们免受垃圾短信骚扰成为当前亟待解决的问题。

1.2 实验要求

1. 任务提供包括数据读取、基础模型、模型训练等基本代码
2. 参赛选手需完成核心模型构建代码，并尽可能将模型调到最佳状态
3. 模型单次推理时间不超过 10 秒

1.3 实验环境

可以使用基于 $Python$ 的 $Pandas$、$Numpy$、$Sklearn$ 等库进行相关特征处理，使用 $Sklearn$框架训练分类器，也可编写深度学习模型，使用过程中请注意 $Python$ 包（库）的版本。

二、设计思想

(1)模型构建与训练

首先，我们导入停用词词库，读取预训练集中的`sms_pub.csv`文件

```
data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
stopwords_path = r'scu_stopwords.txt'

sms = pd.read_csv(data_path, encoding='utf-8')
sms_pos = sms[(sms['label'] == 1)]
sms_neg = sms[(sms['label'] == 0)].sample(frac=1.0)[: len(sms_pos)]
sms = pd.concat([sms_pos, sms_neg], axis=0).sample(frac=1.0)
```

然后，我们读入停用词，并分行输出

```
def read_stopwords(stopwords_path):
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()
    return stopwords

stopwords = read_stopwords(stopwords_path)#读取停用词
```

然后，我们开始训练我们的模型，先将csv文件中的消息与$label$都读入x和y中，然后将总的数据集分成训练集和测试集，按照4：1的比例进行分块。后面，我们设置`pipeline`，利用`TfidfVectorizer`、`MaxAbsScaler`和`ComplementNB`进行创建训练，将预测的结果赋值给`y_pred`。

```
X = np.array(sms.msg_new)
y = np.array(sms.label)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=22,
test_size=0.2)
print("the number of all the datas", X.shape)
print("the number of the train datas", X_train.shape)
print("the number of the test datas", X_test.shape)

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=stopwords, ngram_range=(1,2))),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', ComplementNB()),
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
```

然后，我们测试我们训练的模型，并选择一个最好的模型进行预测。

```
pipeline.fit(X, y)

import joblib
pipeline_path = 'results/pipeline_now_the_best.model'
joblib.dump(pipeline, pipeline_path)
```

(2)利用训练模型进行预测

首先我们读取停用词，然后导入我们测试出最好的模型

```
stopwords_path=r'scu_stopwords.txt'

def read_stopwords(stopwords_path):
    stopwords=[]
    with open(stopwords_path,'r',encoding='utf-8') as f:
        stopwords=f.read()
        stopwords=stopwords.splitlines()
    return stopwords

stopwords=read_stopwords(stopwords_path)
pipeline_path='results/pipeline_now_the_best.model'
pipeline=joblib.load(pipeline_path)
```

然后利用`predict`函数进行预测就可以了，返回$label$值和概率

```
def predict(message):
    label=pipeline.predict([message])[0]
    proba=list(pipeline.predict_proba([message])[0])
    return label,proba
```

三、代码内容

本次实验很明显，我们需要`train.py`和`main.py`两个文件来完成实验。`train.py`文件我们用于训练我们的模型，将训练好的`pipeline.model`存储在`result`文件夹中，在$main$文件中测试我们的十个样例。下面我将两个文件的源代码展示如下：

(1)train.py

```
import os
os.environ["HDF5_USE_FILE_LOCKING"] = "FALSE"
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import ComplementNB
from sklearn.preprocessing import MaxAbsScaler

data_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms_pub.csv"
stopwords_path = r'scu_stopwords.txt'

sms = pd.read_csv(data_path, encoding='utf-8')
sms_pos = sms[(sms['label'] == 1)]
sms_neg = sms[(sms['label'] == 0)].sample(frac=1.0)[: len(sms_pos)]
sms = pd.concat([sms_pos, sms_neg], axis=0).sample(frac=1.0)

def read_stopwords(stopwords_path):
    with open(stopwords_path, 'r', encoding='utf-8') as f:
        stopwords = f.read()
    stopwords = stopwords.splitlines()
    return stopwords
```

```
stopwords = read_stopwords(stopwords_path)#读取停用词

X = np.array(sms.msg_new)
y = np.array(sms.label)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=22,
test_size=0.2)
print("the number of all the datas", X.shape)
print("the number of the train datas", X_train.shape)
print("the number of the test datas", X_test.shape)

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer(stop_words=stopwords, ngram_range=(1,2))),
    ('MaxAbsScaler', MaxAbsScaler()),
    ('classifier', ComplementNB()),
])

pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)

from sklearn.metrics import roc_auc_score
from sklearn import metrics

print("the model's AUV: ", roc_auc_score(y_test, y_pred))
print("在测试集上的混淆矩阵: ")
print(metrics.confusion_matrix(y_test, y_pred))
print("在测试集上的分类结果报告: ")
print(metrics.classification_report(y_test, y_pred))
print("在测试集上的 f1-score : ")
print(metrics.f1_score(y_test, y_pred))

pipeline.fit(X, y)

import joblib
pipeline_path = 'results/pipeline_now_the_best.model'
joblib.dump(pipeline, pipeline_path)
```

(2)main.py

```
import os
import joblib
os.environ["HD5_USE_FILE_LOCKING"]="FALSE"

stopwords_path=r'scu_stopwords.txt'

def read_stopwords(stopwords_path):
    stopwords=[]
    with open(stopwords_path,'r',encoding='utf-8') as f:
        stopwords=f.read()
        stopwords=stopwords.splitlines()
    return stopwords

stopwords=read_stopwords(stopwords_path)
```

```
pipeline_path='results/pipeline_now_the_best.model'
pipeline=joblib.load(pipeline_path)

def predict(message):
    label=pipeline.predict([message])[0]
    proba=list(pipeline.predict_proba([message])[0])
    return label,proba
```

四、实验结果

(1)模型训练

我们将模型训练中最好的模型存储在了`results`文件夹下的`pipeline_now_the_best.model`中，下面是我们测试时的模型各项指标：

Run: Python

```
[4]: !cd $WORK && python 'train.py'


the number of all the datas (158292,)
the number of the train datas (126633,)
the number of the test datas (31659,)
the model's AUV:  0.9627766460118486
在测试集上的混淆矩阵:
[[14754  1068]
 [  110 15727]]
在测试集上的分类结果报告:
              precision    recall  f1-score   support

           0       0.99      0.93      0.96     15822
           1       0.94      0.99      0.96     15837

    accuracy                           0.96     31659
   macro avg       0.96      0.96      0.96     31659
weighted avg       0.96      0.96      0.96     31659

在测试集上的 f1-score :
0.9639004658004413
```

可以发现，模型获得了0.9628的`AUC`，0.9639的`F1-score`，各项指标也非常符合预期。

(2)实例测试

我们在网站中进行测试，得到了10/10的结果。

接口测试

接口测试通过。

用例测试

测试点	状态	时长	结果
测试模型预测结果	✓	10s	通过测试，训练的分类器具备检测恶意短信的能力，分类正确比例:10/10
测试读取停用词库函数结果	✓	12s	read_stopwords 函数返回的类型正确

确认

虽然我们完成了本次实验的样例测试并且获得10/10的成绩，但是对于模型还可以进一步进行调参优化，在后期我也会去尝试一些其他的深度学习模型来优化我的模型。

五、总结

本实验通过采用朴素贝叶斯分类器和`TF-IDF`特征提取方法，实现了对垃圾短信的识别。根据实验结果，该方法在测试集上的`F1-score`和`AUC`上表现良好，达到了预期目标。

当然，我们还可以调整参数来实现模型的优化，也可以选择其他的机器学习、深度学习算法模型来提升与优化我们的模型。

由于上学期选了陈晨老师的python语言程序设计，所以本次实验的代码部分自我感觉难度较低，也进一步了解了NLP。