**垃圾短信识别问题程序报告**

学号2112213姓名：冯思程

1. **问题重述**

垃圾短信 (Spam Messages，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。

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

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。

1. **设计思想**

**方法：**

首先，给出的朴素贝叶斯在垃圾短信识别方向上分类效果足够优秀，已经不需要再去进行模型的更换。那么基于原来给出的训练框架在mo平台上新建train.py文件进行训练，其中要对读取停用词的函数进行补充，代码如下：

def read\_stopwords(stopwords\_path):

"""

读取停用词库

:param stopwords\_path: 停用词库的路径

:return: 停用词列表

"""

stopwords=[]

with open(stopwords\_path, 'r', encoding='utf-8') as f:

stopwords = f.read()

stopwords = stopwords.splitlines()

return stopwords

然后按照示例给的pipeline\_list进行训练，并将model存在指定路径下，这里我重新命名了model为pipeline-new，存完model后，进行一次main.py测试，发现虽然可以成功实现分类目标和读取停用词，但是分类正确比例是8/10，不够高，于是考虑进行优化。

**优化方向：**

在n次的不断调试下，最后找到了一个最终测试结果是10/10的model，这里的我用到TfidfVectorizer模块来进行文本的向量化（TF-IDF方法），然后对其中的ngram参数进行设定，设定为（1，3）来提高切分时候词组的长度来提升对特征信息的获取效率，然后对朴素贝叶斯的分类器的alpha参数进行设定，这里经过不断调试将其设定为0.99，这个参数较大可以有效提升训练模型的泛化能力。这里本来我打算采用归一化，用MaxAbsScaler模块来进行归一化处理，但是发现这种处理之后，测试的10个案例分类效果不好，于是放弃这个优化。综上我的train.py文件的代码如下：

# 导入相关的包

import warnings

warnings.filterwarnings('ignore')

import os

os.environ["HDF5\_USE\_FILE\_LOCKING"] = "FALSE"

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import MaxAbsScaler

from joblib import dump,load

def read\_stopwords(stopwords\_path):

"""

读取停用词库

:param stopwords\_path: 停用词库的路径

:return: 停用词列表

"""

stopwords=[]

with open(stopwords\_path, 'r', encoding='utf-8') as f:

stopwords = f.read()

stopwords = stopwords.splitlines()

return stopwords

# 数据集的路径

data\_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms\_pub.csv"

# 读取数据

sms = pd.read\_csv(data\_path, encoding='utf-8')

# 停用词库路径

stopwords\_path = r'scu\_stopwords.txt'

# 读取停用词

stopwords = read\_stopwords(stopwords\_path)

# 导入数据

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=42, test\_size=0.1)

pipeline\_list = [

# --------------------------- 需要完成的代码 ------------------------------

# ========================== 以下代码仅供参考 =============================

('tfidf', TfidfVectorizer(ngram\_range=(1,3),token\_pattern=r"(?u)\b\w+\b",stop\_words=stopwords)),

('clf', MultinomialNB(alpha=0.99))

# 分类器

# ======================================================================

# ------------------------------------------------------------------------

]

# 搭建 pipeline

pipeline = Pipeline(pipeline\_list)

# 训练 pipeline

pipeline.fit(X\_train, y\_train)

# 对测试集的数据集进行预测

y\_pred = pipeline.predict(X\_test)

# 在测试集上进行评估

from sklearn import metrics

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))

# 在所有的样本上训练一次，充分利用已有的数据，提高模型的泛化能力

pipeline.fit(X, y)

# 保存训练的模型，请将模型保存在 results 目录下

from sklearn.externals import joblib

pipeline\_path = 'results/pipeline-new.model'

joblib.dump(pipeline, pipeline\_path)

然后确定了train.py之后，将main确定，只需添加一下对TfidfVectorizer的import，其他是默认勾选的内容并将读取停用词的函数补充完整。综上完成本次实验任务，而且对message测试的10个案例的分类正确比例达到10/10。

**局限性：**

1.停用词集的不完备，本次实验用的是四川大学机器智能实验室停用词库，如果能对停用词库进行进一步的优化，将可以提升对短信信息的噪声废话的排除效果。

2.对于本次训练采用的朴素贝叶斯方法，我认为还可以进一步优化或者更换其他模型进行调参。

3.我认为本次实验测试结果的10个案例有点太少了，不是很好的反应整个训练模型的泛化能力和准确率，可以加大验证数据集的大小，可以更好的对模型参数、数据处理进行调试优化。

1. **代码内容**

**Train.py**的代码是：

# 导入相关的包

import warnings

warnings.filterwarnings('ignore')

import os

os.environ["HDF5\_USE\_FILE\_LOCKING"] = "FALSE"

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import MaxAbsScaler

from joblib import dump,load

def read\_stopwords(stopwords\_path):

"""

读取停用词库

:param stopwords\_path: 停用词库的路径

:return: 停用词列表

"""

stopwords=[]

with open(stopwords\_path, 'r', encoding='utf-8') as f:

stopwords = f.read()

stopwords = stopwords.splitlines()

return stopwords

# 数据集的路径

data\_path = "./datasets/5f9ae242cae5285cd734b91e-momodel/sms\_pub.csv"

# 读取数据

sms = pd.read\_csv(data\_path, encoding='utf-8')

# 停用词库路径

stopwords\_path = r'scu\_stopwords.txt'

# 读取停用词

stopwords = read\_stopwords(stopwords\_path)

# 导入数据

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=42, test\_size=0.1)

pipeline\_list = [

# --------------------------- 需要完成的代码 ------------------------------

# ========================== 以下代码仅供参考 =============================

('tfidf', TfidfVectorizer(ngram\_range=(1,3),token\_pattern=r"(?u)\b\w+\b",stop\_words=stopwords)),

('clf', MultinomialNB(alpha=0.99))

# 分类器

# ========================================================================

# ------------------------------------------------------------------------

]

# 搭建 pipeline

pipeline = Pipeline(pipeline\_list)

# 训练 pipeline

pipeline.fit(X\_train, y\_train)

# 对测试集的数据集进行预测

y\_pred = pipeline.predict(X\_test)

# 在测试集上进行评估

from sklearn import metrics

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))

# 在所有的样本上训练一次，充分利用已有的数据，提高模型的泛化能力

pipeline.fit(X, y)

# 保存训练的模型，请将模型保存在 results 目录下

from sklearn.externals import joblib

pipeline\_path = 'results/pipeline-new.model'

joblib.dump(pipeline, pipeline\_path)

**Main.py**的代码是：

import os

os.environ["HDF5\_USE\_FILE\_LOCKING"] = "FALSE"

# ---------- 停用词库路径，若有变化请修改 -------------

stopwords\_path = r'scu\_stopwords.txt'

# ---------------------------------------------------

def read\_stopwords(stopwords\_path):

"""

读取停用词库

:param stopwords\_path: 停用词库的路径

:return: 停用词列表，如 ['嘿', '很', '乎', '会', '或']

"""

stopwords = []

# ----------- 请完成读取停用词的代码 ------------

with open(stopwords\_path, 'r', encoding='utf-8') as f:

stopwords = f.read()

stopwords = stopwords.splitlines()

#----------------------------------------------

return stopwords

# 读取停用词

stopwords = read\_stopwords(stopwords\_path)

# ----------------- 导入相关的库 -----------------

from sklearn.pipeline import Pipeline

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.naive\_bayes import BernoulliNB

from sklearn.naive\_bayes import MultinomialNB

from sklearn.naive\_bayes import ComplementNB

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.preprocessing import MaxAbsScaler

from sklearn.preprocessing import StandardScaler

# 加载训练好的模型

from sklearn.externals import joblib

# ------- pipeline 保存的路径，若有变化请修改 --------

pipeline\_path = 'results/pipeline-new.model'

# --------------------------------------------------

pipeline = joblib.load(pipeline\_path)

def predict(message):

"""

预测短信短信的类别和每个类别的概率

param: message: 经过jieba分词的短信，如"医生 拿 着 我 的 报告单 说 ： 幸亏 你 来 的 早 啊"

return: label: 整数类型，短信的类别，0 代表正常，1 代表恶意

proba: 列表类型，短信属于每个类别的概率，如[0.3, 0.7]，认为短信属于 0 的概率为 0.3，属于 1 的概率为 0.7

"""

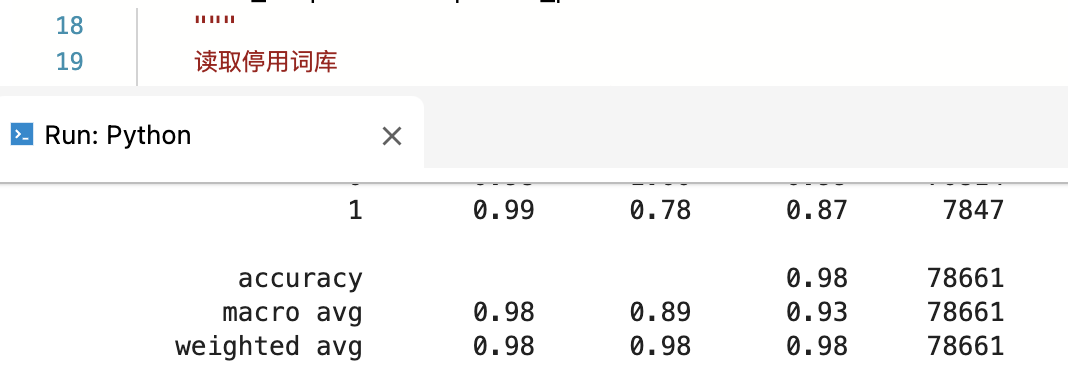
label = pipeline.predict([message])[0]

proba = list(pipeline.predict\_proba([message])[0])

return label, proba

1. **实验结果**

运行train.py进行一下在测试集上进行一下验证，准确率到达了0.98，是一个较为有效的分类器：



在Mo环境下进行代码运行测试，运行结果如下：



在第46次调试中到达了对于给的示例短信集分类正确比例100%的突破。

1. **总结**

在这次实验中，我学习到了如何使用朴素贝叶斯分类器来对垃圾短信进行识别。在构建模型的过程中，我使用了TfidfVectorizer来对文本进行向量化，并使用了Pipeline来将分类器和向量化器结合在一起，从而简化了模型训练的流程。此外，我还学习了如何使用train\_test\_split函数对数据集进行分割，并对模型的性能进行评估。通过不断调试模型参数，我得到了一个能够很好地对垃圾短信进行分类的模型。

在这个过程中，我也遇到了一些问题。比如，在调试模型时，需要根据实际情况选择合适的参数，否则可能会出现过拟合或欠拟合等问题。此外，在进行文本向量化时，选择合适的向量化器和相关参数也非常关键，因为它们直接影响了模型的性能和效果。

总之，这次实验让我对朴素贝叶斯分类器和文本分类有了更深入的了解，同时也提高了我的编程能力和数据分析能力。在未来的工作中，我将继续深入学习和掌握相关技术，不断提升自己的能力。

同时我也对深度学习这种可以自动学习特征的方法有了更加深刻的了解，提升了我对其研究的兴趣。