**《大数据智能》**

**（ 2020学年 秋季 学期）**

**作**

**业**

**报**

**告**

**学 号： 1180400510**

**姓 名： 石瑞河**

**班 级： 1804105**

**任课教师： 朱聪慧**

作业报告

实验名称： 构建中文文本分类系统

成绩：

实验类别：验证/综合型实验 实验要求：1人1组 时间：

## ￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣￣

1. 实验目的

熟悉词袋模型，使用sklearn框架中的贝叶斯或者SVM分类器，训练分类模型（在训练数据上），测试分类性能（在测试数据上）。

1. 实验内容

* 文件格式处理：

1. 解压文本分类数据包“文本分类.zip”。

C000008 文件夹 --财经 类别

C000024 文件夹 --军事 类别

每个文件是一个句子，已经分词，去掉停用词

文件号0~200，用作训练集

文件号201~220，用作测试集

1. 在所有文件，统计词频（可以合并，打开总的，也可以用程序遍历文件夹，打开每一个文件）
2. 将输入文件表示为标准词袋模型（分别生成训练、测试文件）
3. 训练模型（svm或者贝叶斯）
4. 测试原始词袋模型性能（使用精确率指标）
5. 加入TF-IDF特征（重新生成训练、测试文件）
6. 训练模型（svm或者贝叶斯）
7. 测试融合TF-IDF词袋模型性能（使用精确率指标）

三、使用的算法描述

SVM（支持向量机）是一种常用的分类算法，SVM通过核函数将训练数据映射道高维，并在高维之中寻找一个最优超平面来将数据尽可能的正确分类，且到超平面距离尽可能远，常用的核函数如线性核、高斯核等。

词袋模型是将一个文件用向量来表示，向量每个位置表示文件中是否出现对应词。

TF-IDF为词频率和逆文档频率的乘积，在词袋模型中加入TF-IDF特征，是加入了词语出现频率和词语的独特性（是否与某类比高度关联）考量，是很有意义的分类信息

1. 程序源码（拷贝至此处，或者作为附件和报告一起提交均可）

import numpy as np

from collections import Counter

from sklearn.feature\_extraction.text import CountVectorizer, TfidfTransformer

from sklearn.metrics import precision\_score

from sklearn.svm import SVC

data\_train = []

data\_test = []

word\_list = []

#读取文本,生成训练集和测试集,统计词频

for pack in ("08","24"):

    dir\_path = "data/C0000" + pack + '/'

    for i in range (221):

        path = dir\_path + str(i) + ".cut"

        with open(path, "r",encoding = "utf-8") as f:

            line = f.readline()

            if i <= 200 :

                data\_train.append(line)

            else :

                data\_test.append(line)

            temp = line.split()

            for word in temp : word\_list.append(word)

        f.close()

test\_label = ["财经" for \_ in range(20)] + ["军事" for \_ in range(20)]

train\_label = ["财经" for \_ in range(201)] + ["军事" for \_ in range(201)]

word\_count = Counter(word\_list)

print(word\_count.most\_common(10))

#词袋模型

wordbag = CountVectorizer()

train\_vec = wordbag.fit\_transform(data\_train)

test\_vec = wordbag.transform(data\_test)

svm = SVC().fit(train\_vec, train\_label)

svm\_predict = svm.predict(test\_vec)

score = precision\_score(test\_label, svm\_predict, average='macro')

print("原始词袋性能: ",score)

#TF-IDF

tfidf = TfidfTransformer()

train\_vec\_tfidf = tfidf.fit\_transform(train\_vec)

test\_vec\_tfidf = tfidf.transform(test\_vec)

svm = SVC().fit(train\_vec\_tfidf, train\_label)

svm\_predict = svm.predict(test\_vec\_tfidf)

score = precision\_score(test\_label, svm\_predict, average='macro')

print("TF-IDF词袋性能: ",score)

五、程序运行结果（将程序运行结果的截图，拷贝至此处）  


六、心得体会

词袋模型的精确率为85.35%，加入TF-IDF特征后精确率提升到97.62%，效果提升明显

Python某种意义上是一门吃经验的语言，看得多才会得多