山东大学＿＿＿计算机科学与技术＿＿＿学院

＿信息检索与数据挖掘＿课程实验报告

学号：201600301304	姓名：贾乘兴	班级：人工智能 16
实验题目：朴素贝叶斯分类器（多分类）		

实验内容：

基于向量空间模型的部分方法实现二十分类的朴素贝叶斯分类器

朴素贝叶斯基于的公式为贝叶斯公式，即：

$$p(c_i|x)=\frac{p(x|c_i)p(c_i)}{p(x,c)}$$

其中 ci 为文本类别，x 为文本，朴素贝叶斯分类器基于假设为各单词之间的顺序对结果无影响，即若文本 x 中有词 x1，x2，x3，x4……，则 p（x|ci）=p（x1|ci）p（x2|ci）p（x3|ci）p（x4|ci）……。虽然假设较为简单，但分类效果较好。

本次实验使用的是二十分类数据集，取 1/9 为测试集，8/9 为训练集，最终二十分类正确率 80%，且大多数数据的 precision 都达到了 90%以上，代码如下：

```
import os
import chardet
import re
import nltk
import numpy
import math

# 去除停用词
def cutstopwords(str):
    stopwords = {}.fromkeys([line.rstrip() for line in
open('estopwords.txt')])
    segs = str.replace('\n','').lower().split(' ')
    new_str = ''
    for seg in segs:
        if seg not in stopwords:
            new_str = new_str + " " +seg
    return new_str

# 去除标点
def cutsyms(str):
    new_str = re.sub('[1234567890,.\'\"\t\n*_+=?/|!@#$%^&*()`~<>:;\-
\[\]]'," ",str)
    return new_str
```

```
# 词干提取
def stemming(str):
    s = nltk.stem.SnowballStemmer('english')
    segs = str.replace('\n', '').lower().split(' ')
    new_str = ''
    for seg in segs:
        new_str = new_str + " " + s.stem(seg)
    return new_str

# 读取文本
def readtxt(path):
    global class_dict
    global class_num
    global class_list
    global num_txt
    global num_dict
    num_dict = {}
    num_txt = 0
    all_context = ""
    for dirName, subdirList, fileList in os.walk(path):
        fileList.remove(fileList[0])
        for fname in fileList:
            class_name = dirName.split('/')[6]
            num = class_list[class_name]
            class_num[num] = class_num[num] + 1

            fname = os.path.join(dirName, fname)
            f = open(fname, 'rb')
            data = f.read()
            f.close()

            print(chardet.detect(data))
            print(fname)

            fname =
open(fname,'r+',encoding=chardet.detect(data)['encoding'])
            str = fname.read()
            str = cutsyms(str)
            str = cutstopwords(str)
            str = stemming(str)

            str_list = str.replace('\n', '').lower().split(' ')
            for seg in str_list:
                if seg in num_dict.keys():
                    num_dict[seg] = num_dict[seg] + 1
                else:
                    num_dict[seg] = 1

            for seg in str_list:
                if seg in class_dict[class_name].keys():

class_dict[class_name].update({seg:class_dict[class_name][seg] + 1})
                else:
                    class_dict[class_name].update({seg:1})

            all_context = all_context + "\n" + str
            num_txt = num_txt + 1
            fname.close()
    return all_context

# 统计词出现次数
def wordcount(str):
    strl_ist = str.replace('\n','').lower().split(' ')
    count_dict = {}
    for str in strl_ist:
        if str in count_dict.keys():
            count_dict[str] = count_dict[str] + 1
        else:
            count_dict[str] = 1
    # count_list=sorted(count_dict.items(),key=lambda x:x[1],reverse=True)
    count_dict.pop('')
    return count_dict

#全部文本读取
num_txt = 0
num_dict = {}
class_list = {'alt.atheism':0,'comp.graphics':1,'comp.os.ms-
windows.misc':2,'comp.sys.ibm.pc.hardware':3,'comp.sys.mac.hardware':4,'co
mp.windows.x':5,'misc.forsale':6,'rec.autos':7,'rec.motorcycles':8,'rec.sp
ort.baseball':9,'rec.sport.hockey':10,'sci.crypt':11,'sci.electronics':12,
'sci.med':13,'sci.space':14,'soc.religion.christian':15,'talk.politics.gun
s':16,'talk.politics.mideast':17,'talk.politics.misc':18,'talk.religion.mi
sc':19}
class_num = [0]*20
class_dict = {'alt.atheism':{},'comp.graphics':{},'comp.os.ms-
windows.misc':{},'comp.sys.ibm.pc.hardware':{},'comp.sys.mac.hardware':{},
'comp.windows.x':{},'misc.forsale':{},'rec.autos':{},'rec.motorcycles':{},
'rec.sport.baseball':{},'rec.sport.hockey':{},'sci.crypt':{},'sci.electronics':{},'sci.med':{},'sci.space':{},'soc.religion.christian':{},'talk.politics.guns':{},'talk.politics.mideast':{},'talk.politics.misc':{},'talk.religion.misc':{}}
context = readtxt("/Users/apple/Desktop/ir/train")

#全部文档记数，去除高频低频词
str_dict = wordcount(context)
new_dict = str_dict
str_dict = {}
for seg in new_dict:
    if (new_dict[seg] > 10) & (new_dict[seg] < 10000):
        str_dict[seg] = new_dict[seg]
if '' in str_dict.keys():
    str_dict.pop('')
length = len(str_dict)
print(length)

for i in class_list.keys():
    z_num = class_list[i]
    newc_dict = {}
    for seg in str_dict:
        if seg in class_dict[i].keys():
            newc_dict[seg] = class_dict[i][seg]
    class_dict[i] = newc_dict

class_sum = [0]*20
for i in class_list.keys():
    x_num = class_list[i]
    for j in class_dict[i].keys():
        class_sum[x_num] = class_sum[x_num] + class_dict[i][j]

#classification
classification = [[0 for x in range(20)]for y in range(20)]
for dirName, subdirList, fileList in os.walk('/Users/apple/Desktop/ir/test'):
    fileList.remove(fileList[0])
    for fname in fileList:
        name = dirName.split('/')[6]
        num = class_list[name]
        print(name)

        fname = os.path.join(dirName, fname)
        f = open(fname, 'rb')
        data = f.read()
        f.close()

        fname = open(fname, 'r+',
encoding=chardet.detect(data)['encoding'])
        ins_str = fname.read()
        fname.close()

        ins_dict = {}
        ins_str = cutsyms(ins_str)
        ins_str = cutstopwords(ins_str)
        ins_str = stemming(ins_str)
        ins_dict = wordcount(ins_str)

        class_pro = [0]*20
        for i in class_list.keys():
            y_num = class_list[i]
            for seg in ins_dict.keys():
                if seg in class_dict[i].keys():
                    class_pro[y_num] = class_pro[y_num] +
math.log((class_dict[i][seg]+1)/(class_sum[y_num]+20))
                else:
                    class_pro[y_num] = class_pro[y_num] +
math.log((1)/(class_sum[y_num]+20))
            class_pro[y_num] = class_pro[y_num] +
math.log((class_num[y_num])/(2000))

        index = numpy.argmax(class_pro)
        classification[num][index] = classification[num][index] + 1

sum = 0
for i in range(20):
    sum = sum + classification[i][i]
print(classification)
print(sum/2000)
```

评价标准：召回率与准确率，可以进行综合考虑，取调和均值，本实验可以得到 20*20 的矩阵，便于后续处理

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

1. 第三类文本的分类效果不好：分析由于文本较短，故所得词汇较少，可以在预处理时对去除的文本进行改进

结论分析与体会： 朴素贝叶斯虽然简单，但效果较好，很多情况下基于的假设虽然完全正确，但可以解决实际问题