import jieba.posseg as psg

import xlrd

import xlwt

title=str('创新成果')

ExcelFile=xlrd.open\_workbook(title+".xls")

sheet=ExcelFile.sheet\_by\_name('wordCount')

workbook = xlwt.Workbook(encoding = 'ascii')

worksheet = workbook.add\_sheet('My Worksheet')

rows\_num = sheet.nrows

for i in range (1,rows\_num):

temp = str(sheet.cell(i,0).value)

worksheet.write(i, 0, label = temp)

temp =psg.cut(temp)

count=0

for w in temp:

count=count+1

#print(w.flag)

worksheet.write(i, count, label = str(w.flag))

workbook.save(title+"1.xls")

from snownlp import SnowNLP

import xlrd

import xlwt

name=["体育类实践活动","创新成果","游学经历","社团活动","艺术成果展示","艺术类实践活动"]

for j in range(0,6):

tempname="附中G17级2018.2-9 指定维度的记录数据/"+name[j]+".xlsx"

ExcelFile=xlrd.open\_workbook(tempname)

sheet=ExcelFile.sheet\_by\_name('Sheet1')

workbook = xlwt.Workbook(encoding = 'ascii')

worksheet = workbook.add\_sheet('My Worksheet')

rows\_num = sheet.nrows

for i in range (1,rows\_num):

temp = str(sheet.cell(i,0).value)

s = SnowNLP(temp)

sentiment=s.sentiments

worksheet.write(i, 0, label = str(sentiment))

tempname="附中G17级2018.2-9 指定维度的记录数据/情感/"+name[j]+"Excel\_Workbook.xls"

workbook.save(tempname)

#!/usr/bin/python

# -\*- coding:utf-8 -\*-

import sys

import importlib

importlib.reload(sys)

import jieba

import jieba.analyse

import xlwt #写入Excel表的库

title=str('艺术实践')

if \_\_name\_\_=="\_\_main\_\_":

wbk = xlwt.Workbook(encoding = 'ascii')

sheet = wbk.add\_sheet("wordCount")#Excel单元格名字

word\_lst = []

key\_list=[]

for line in open('1.txt'):#1.txt是需要分词统计的文档

item = line.strip('\n\r').split('\t') #制表格切分

# print item

tags = jieba.analyse.extract\_tags(item[0]) #jieba分词

for t in tags:

word\_lst.append(t)

word\_dict= {}

with open(title+".txt",'w') as wf2: #打开文件

for item in word\_lst:

if item not in word\_dict: #统计数量

word\_dict[item] = 1

else:

word\_dict[item] += 1

orderList=list(word\_dict.values())

orderList.sort(reverse=True)

# print orderList

for i in range(len(orderList)):

for key in word\_dict:

if word\_dict[key]==orderList[i]:

wf2.write(key+' '+str(word\_dict[key])+'\n') #写入txt文档

key\_list.append(key)

word\_dict[key]=0

for i in range(len(key\_list)):

sheet.write(i, 1, label = orderList[i])

sheet.write(i, 0, label = key\_list[i])

wbk.save(title+'.xls') #保存为 wordCount.xls文件

import jieba

import numpy as np

import xlrd

import xlwt

def open\_dict(Dict = 'hahah', path=r'D:\学习\学习\高中\学校事情\高二\高研\论文\appendix\Textming'):

path = path + '%s.txt' % Dict

dictionary = open(path, 'r', encoding='utf-8')

dict = []

for word in dictionary:

word = word.strip('\n')

dict.append(word)

return dict

def judgeodd(num):

if (num % 2) == 0:

return 'even'

else:

return 'odd'

deny\_word = open\_dict(Dict = 'deny\_word', path= r'')

posdict = open\_dict(Dict = 'positive', path= r'')

negdict = open\_dict(Dict = 'negative', path= r'')

degree\_word = open\_dict(Dict = 'degree\_word', path= r'')

mostdict = degree\_word[degree\_word.index('extreme')+1 : degree\_word.index('very')]

verydict = degree\_word[degree\_word.index('very')+1 : degree\_word.index('more')]

moredict = degree\_word[degree\_word.index('more')+1 : degree\_word.index('ish')]

ishdict = degree\_word[degree\_word.index('ish')+1 : degree\_word.index('last')]

def sentiment\_score\_list(dataset):

seg\_sentence = dataset.split('。')

count1 = []

count2 = []

for sen in seg\_sentence:

segtmp = jieba.lcut(sen, cut\_all=False)

i = 0

a = 0

poscount = 0

poscount2 = 0

poscount3 = 0

negcount = 0

negcount2 = 0

negcount3 = 0

for word in segtmp:

if word in posdict:

poscount += 1

c = 0

for w in segtmp[a:i]:

if w in mostdict:

poscount \*= 4.0

elif w in verydict:

poscount \*= 3.0

elif w in moredict:

poscount \*= 2.0

elif w in ishdict:

poscount \*= 0.5

elif w in deny\_word:

c += 1

if judgeodd(c) == 'odd':

poscount \*= -1.0

poscount2 += poscount

poscount = 0

poscount3 = poscount + poscount2 + poscount3

poscount2 = 0

else:

poscount3 = poscount + poscount2 + poscount3

poscount = 0

a = i + 1

elif word in negdict:

negcount += 1

d = 0

for w in segtmp[a:i]:

if w in mostdict:

negcount \*= 4.0

elif w in verydict:

negcount \*= 3.0

elif w in moredict:

negcount \*= 2.0

elif w in ishdict:

negcount \*= 0.5

elif w in degree\_word:

d += 1

if judgeodd(d) == 'odd':

negcount \*= -1.0

negcount2 += negcount

negcount = 0

negcount3 = negcount + negcount2 + negcount3

negcount2 = 0

else:

negcount3 = negcount + negcount2 + negcount3

negcount = 0

a = i + 1

elif word == '！' or word == '!':

for w2 in segtmp[::-1]:

if w2 in posdict or negdict:

poscount3 += 2

negcount3 += 2

break

i += 1

pos\_count = 0

neg\_count = 0

if poscount3 < 0 and negcount3 > 0:

neg\_count += negcount3 - poscount3

pos\_count = 0

elif negcount3 < 0 and poscount3 > 0:

pos\_count = poscount3 - negcount3

neg\_count = 0

elif poscount3 < 0 and negcount3 < 0:

neg\_count = -poscount3

pos\_count = -negcount3

else:

pos\_count = poscount3

neg\_count = negcount3

count1.append([pos\_count, neg\_count])

count2.append(count1)

count1 = []

return count2

def sentiment\_score(senti\_score\_list):

score = []

for review in senti\_score\_list:

score\_array = np.array(review)

Pos = np.sum(score\_array[:, 0])

Neg = np.sum(score\_array[:, 1])

AvgPos = np.mean(score\_array[:, 0])

AvgPos = float('%.1f'%AvgPos)

AvgNeg = np.mean(score\_array[:, 1])

AvgNeg = float('%.1f'%AvgNeg)

StdPos = np.std(score\_array[:, 0])

StdPos = float('%.1f'%StdPos)

StdNeg = np.std(score\_array[:, 1])

StdNeg = float('%.1f'%StdNeg)

score.append([Pos, Neg, AvgPos, AvgNeg, StdPos, StdNeg])

return score

name=["体育类实践活动","创新成果","游学经历","社团活动","艺术成果展示","艺术类实践活动"]

for j in range(0,6):

tempname="内容/"+name[j]+".xlsx"

ExcelFile=xlrd.open\_workbook(tempname)

sheet=ExcelFile.sheet\_by\_name('Sheet1')

workbook = xlwt.Workbook(encoding = 'ascii')

worksheet = workbook.add\_sheet('My Worksheet')

rows\_num = sheet.nrows

for i in range (1,rows\_num):

temp = str(sheet.cell(i,0).value)

#print(i)

s = sentiment\_score(sentiment\_score\_list(temp))

length=len(s[0])

positiv=0

negativ=0

for k in range (0,length,2):

positiv=positiv+s[0][k]

for k in range (1,length+1,2):

negativ=negativ+s[0][k]

worksheet.write(i, 0, label = str(positiv))

worksheet.write(i, 1, label = str(negativ))

tempname="内容/正负分/"+name[j]+"Excel\_Workbook.xls"

workbook.save(tempname)

# -\*- coding: utf-8 -\*-

\_\_author\_\_ = 'leilu'

#wordcloud生成中文词云

from wordcloud import WordCloud

import codecs

import jieba

#import jieba.analyse as analyse

from scipy.misc import imread

import os

from os import path

import matplotlib.pyplot as plt

from PIL import Image, ImageDraw, ImageFont

# 绘制词云

def draw\_wordcloud():

#读入一个txt文件

comment\_text = open('out6.txt','r').read()

#结巴分词，生成字符串，如果不通过分词，无法直接生成正确的中文词云

cut\_text = " ".join(jieba.cut(comment\_text))

d = path.dirname(\_\_file\_\_) # 当前文件文件夹所在目录

color\_mask = imread("alice.png") # 读取背景图片

cloud = WordCloud(

#设置字体，不指定就会出现乱码

font\_path="simkai.ttf",

#font\_path=path.join(d,'simsun.ttc'),

#设置背景色

background\_color='white',

#词云形状

mask=color\_mask,

#允许最大词汇

max\_words=2000,

#最大号字体

max\_font\_size=80

)

word\_cloud = cloud.generate(cut\_text) # 产生词云

plt.savefig("pjl\_cloud4.jpg",dpi=4000)

#word\_cloud.to\_file("pjl\_cloud4.jpg") #保存图片

# 显示词云图片

plt.imshow(word\_cloud)

plt.axis('off')

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

draw\_wordcloud()

#encoding=utf-8

# \_\*\_ coding:utf-8 \_\*\_

import time

from sklearn.metrics import precision\_recall\_fscore\_support

import numpy as np

from sklearn import cross\_validation

from sklearn import metrics

import jieba

from jieba import analyse

jieba.load\_userdict("./data/ntusd-negative.txt")

jieba.load\_userdict("./data/neg\_basic.txt")

jieba.load\_userdict("./data/ntusd-positive.txt")

jieba.load\_userdict("./data/pos\_basic.txt")

def timeDecor(func):

def innerDef(\*args, \*\*kwargs):

t1 = time.time()

result = func(\*args, \*\*kwargs)

t2 = time.time()

t = t2-t1

print ("{0}函数部分运行时间：{1}s".format(str(func.\_\_name\_\_),t)) #下划线是两个

return result

return innerDef

# 支持向量机SVM分类器

@timeDecor

def svm\_classify(X\_train, y\_train, X\_test, y\_test):

from sklearn import svm

print ("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n SVM\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

# param\_grid = {

# # 'C': [1e3, 5e3, 1e4, 5e4, 1e5],

# 'kernel': ['rbf', 'linear', 'poly', 'sigmoid']

# # 'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1],

# }

t0 = time.time()

clf = svm.SVC()

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

print("svm done in %0.3fs" % (time.time()-t0))

pre\_y\_test = clf.predict(X\_test)

print("SVM Metrics: {0}".format(precision\_recall\_fscore\_support(y\_test, pre\_y\_test)))

print(metrics.classification\_report(y\_test, pre\_y\_test))

print("accurary\t"+str(np.mean(pre\_y\_test == y\_test)))

# 朴素贝叶斯分类器

@timeDecor

def nb\_classify(X\_train, y\_train, X\_test, y\_test):

from sklearn.naive\_bayes import GaussianNB

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n NB\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

t0 = time.time()

clf = GaussianNB()

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

print("nb done in %0.3fs" % (time.time()-t0))

pre\_y\_test = clf.predict(X\_test)

print("nb Metrics: {0}".format(precision\_recall\_fscore\_support(y\_test, pre\_y\_test)))

print(metrics.classification\_report(y\_test, pre\_y\_test))

print("accurary\t"+str(np.mean(pre\_y\_test == y\_test)))

def extractFeatures():

tpk = 250

rt = 1.5

path = './data/'

file1 = open(path+'pos\_'+ strs +'.txt', 'rb')

data = file1.read()

w1={}

for w in sorted(analyse.extract\_tags(data,withWeight=True,topK=tpk),key=lambda d:d[1], reverse=True):

w1[w[0]] = w[1]

file2 = open(path+'neg\_'+ strs +'.txt', 'rb')

data = file2.read()

w2={}

for w in sorted(analyse.extract\_tags(data,withWeight=True,topK=tpk),key=lambda d:d[1],reverse=True):

w2[w[0]]=w[1]

wtp=[]

for w in w1:

if not w in w2:

pass

else:

f=True

if w2[w]\*rt>w1[w]:

f=False

if f:

wtp.append(w)

w11=wtp

wtp=[]

for w in w2:

if not w in w1:

pass

else:

f=True

if w in w1:

if w1[w]\*rt>w2[w]:

f=False

if f:

wtp.append(w)

w22=wtp

tow = open(path+'keywords\_'+ strs +'.txt', 'wb')

for w in w11:

tow.write(w.encode("utf-8")+b" ")

for w in w22:

tow.write(w.encode("utf-8")+b" ")

# strs = "hotel"

strs = "book"

# strs = "elec"

extractFeatures()

worddt={}

for line in open("./data/ntusd-positive.txt",encoding='UTF-8'):

worddt[line.strip()] = 0.5

for line in open("./data/ntusd-negative.txt",encoding='UTF-8'):

worddt[line.strip()] = -0.5

for line in open("./data/pos\_basic.txt",encoding='UTF-8'):

worddt[line.strip()] = 0.5

for line in open("./data/neg\_basic.txt",encoding='UTF-8'):

worddt[line.strip()] = -0.5

linenum = 0

for line in open("./data/keywords\_"+ strs +".txt",encoding='UTF-8'):

linenum+=1

if linenum == 1:

for w in line.split():

worddt[w] = 0.5

else:

for w in line.split():

worddt[w] = -0.5

onehotlen=len(worddt)

wordindex={}

for i,key in enumerate(worddt):

wordindex[key]=i

keyword\_feature=[]

onehotfeature=[]

onehotvalue=[]

c = 0

for i,line in enumerate(open("./data/pos\_"+ strs +".txt",encoding='UTF-8')):

temp=[0]\*(onehotlen+2)

for word in jieba.cut(line.strip()):

if word in worddt:

c+=1

temp[wordindex[word]] = 1

if worddt[word] > 0:

temp[-2] += 1

elif worddt[word] < 0:

temp[-1] += 1

onehotfeature.append(temp[:])

onehotvalue.append(1)

if i>1000:

break

for i,line in enumerate(open("./data/neg\_"+ strs +".txt",encoding='UTF-8')):

temp = [0] \* (onehotlen + 2)

for word in jieba.cut(line.strip()):

if word in worddt:

c+=1

temp[wordindex[word]] = 1

if worddt[word] > 0:

temp[-2] += 1

elif worddt[word] < 0:

temp[-1] += 1

onehotfeature.append(temp[:])

onehotvalue.append(-1)

if i>1000:

break

X = onehotfeature

y = onehotvalue

X\_train, X\_test, y\_train, y\_test = cross\_validation.train\_test\_split(X, y, test\_size=0.2, random\_state=0)

svm\_classify(X\_train, y\_train, X\_test, y\_test)

nb\_classify(X\_train, y\_train, X\_test, y\_test)