import math

import os

from collections import Counter

import pickle

import jieba

from SpamMail import settings

def preprocess():

# 对邮件进行分词

with open(os.path.join(settings.BASE\_DIR, "data\_raw.txt"), "r", encoding="utf8") as f:

with open(os.path.join(settings.BASE\_DIR, "data.txt"), "w", encoding="utf8") as fw:

for line in f:

l\_s = line.strip().split("\t")

tokens = jieba.cut(l\_s[1])

s\_split = ' '.join(tokens)

fw.write("{}\t{}\n".format(l\_s[0], s\_split))

def vocabulary():

# 获取邮件中的关键词表，作为特征

stopwords = set()

spam\_count = 0

ham\_count = 0

# 去除停用词，即常见的词和标点符号等

with open(os.path.join(settings.BASE\_DIR, "stop.txt"), "r", encoding="utf-8") as f:

for line in f:

w = line.strip()

stopwords.add(w)

# 统计词频

c\_spam = {}

c\_ham = {}

with open(os.path.join(settings.BASE\_DIR, "data.txt"), "r", encoding="utf8") as f:

for line in f:

\_lines = line.split("\t")

l, s = int(\_lines[0]), str(\_lines[1]).strip()

if l == 1:

spam\_count += 1

for x in s.split():

if x in stopwords:

continue

if x in c\_spam:

c\_spam[x] += 1

else:

c\_spam[x] = 1

else:

ham\_count += 1

for x in s.split():

if x in stopwords:

continue

if x in c\_ham:

c\_ham[x] += 1

else:

c\_ham[x] = 1

one\_vocab = [word for word in c\_spam if c\_spam[word] >= 200]

zero\_vocab = [word for word in c\_ham if c\_ham[word] >= 200]

vocab = set(one\_vocab) | set(zero\_vocab)

print("特征词：", len(vocab), "个")

return vocab, c\_spam, c\_ham, spam\_count, ham\_count

def bayes(vocab, ham\_counter, spam\_counter, ham\_count, spam\_count):

# 写入统计文件，作为贝叶斯分类的特征，直接保存成一个 dict 格式

# 首先需要统计每一个词在垃圾、非垃圾邮件中出现的概率

# p(w|Spam) = w 在垃圾邮件中出现的次数 / 所有垃圾邮件中出现的词数 = w\_spam / vocab\_spam

# p(w|Ham) = w 在正常邮件中出现的次数 / 所有正常邮件中出现的词数 = w\_ham / vocab\_ham

# 统计分母

spam\_vocab = 0

ham\_vocab = 0

# 为了防止出现 0 的情况，加入平滑：

for w in vocab:

w\_spam = spam\_counter[w] + 1

w\_ham = ham\_counter[w] + 1

spam\_vocab += (w\_spam + 1)

ham\_vocab += (w\_ham + 1)

# 计算两个概率

# bayes\_dict[w][0] = p(w|Ham) = w\_ham / ham\_vocab

# bayes\_dict[w][1] = p(w|Spam) = w\_spam / spam\_vocab

bayes\_dict = {}

# 为了防止预测时概率相乘下溢出，改为取对数相加的方式，这里直接保存对数值

for w in vocab:

w\_dict = {}

w\_dict[0] = math.log((ham\_counter[w] + 1) / ham\_vocab)

w\_dict[1] = math.log((spam\_counter[w] + 1) / spam\_vocab)

bayes\_dict[w] = w\_dict

total\_count = {}

total\_count[0] = math.log(ham\_count / (ham\_count + spam\_count))

total\_count[1] = math.log(spam\_count / (ham\_count + spam\_count))

bayes\_dict["\_reserved"] = total\_count

with open(os.path.join(settings.BASE\_DIR, "bayes.pickle"), "wb") as f:

pickle.dump(bayes\_dict, f, pickle.HIGHEST\_PROTOCOL)

def load\_bayes():

with open(os.path.join(settings.BASE\_DIR, "bayes.pickle"), "rb") as f:

bayes\_dict = pickle.load(f)

return bayes\_dict

def test\_bayes(bayes\_dict, sentence):

# 先验概率：

ham\_log\_prob = bayes\_dict["\_reserved"][0]

spam\_log\_prob = bayes\_dict["\_reserved"][1]

for x in jieba.cut(sentence):

if x in bayes\_dict:

ham\_log\_prob += bayes\_dict[x][0]

spam\_log\_prob += bayes\_dict[x][1]

# 哪个大即为哪个类

return 0 if ham\_log\_prob > spam\_log\_prob else 1

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

# 1. 预处理：分词，比较耗时

# c\_spam, c\_ham = preprocess()

# 2. 学习贝叶斯分类器模型参数

vocab, c\_spam, c\_ham, n\_spam, n\_ham = vocabulary()

spam\_counter = Counter(c\_spam)

ham\_counter = Counter(c\_ham)

bayes(vocab, ham\_counter, spam\_counter, n\_ham, n\_spam)

# 3. 加载贝叶斯模型

bayes\_dict = load\_bayes()

# 写入一个可读的文件

with open(os.path.join(settings.BASE\_DIR, "bayes.txt"), "w", encoding="utf8") as f:

for k in bayes\_dict:

f.write("{} {} {}\n".format(k, bayes\_dict[k][0], bayes\_dict[k][1]))

x = test\_bayes(bayes\_dict, "物业与居民的管理")

print(x)

x = test\_bayes(bayes\_dict, "物业处理利用python处理居民的事情")

print(x)

</div>