Казанский (Приволжский) федеральный университет

Институт Вычислительной математики и информационных технологий

Кафедра системного анализа и информационных технологий  
  
  
  
  
  
  
  
Отчет по учебной практике

(Технологическая практика)

Студент

гр.09-641 Валеев А.М.

Руководитель

ассистент кафедры САИТ Нигматуллин Р. Р.

Казань - 2018

**Индивидуальное задание на практику**

Задание практики заключается в разработке фильтра спама для электронных сообщений на основе наивного байесовского классификатора. Для реализации фильтра спама предлагается использовать язык программирования Python 3 и библиотеки Numpy, Scipy, Matplotlib.

**1.Постановка задачи, описание классификатора:**

Наивный байесовский классификатор спама – фильтр спама, для сортировки сообщений. Разделяет поступающие текстовые файлы на сообщения типа HAM и SPAM. В основе классификатора лежит использование теоремы Байеса.

**2.Описание выборки данных для классификации:**

В качестве данных для классификации использовалась база сообщений ENRON - американской энергетической компании, [обанкротившейся](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%BD%D0%BA%D1%80%D0%BE%D1%82%D1%81%D1%82%D0%B2%D0%BE) в [2001 году](https://ru.wikipedia.org/wiki/2001_%D0%B3%D0%BE%D0%B4). В базе хранится 33702 сообщения. В названии каждого сообщения обозначено, к какому классу оно относится.

**3.Описание применяемых структур данных для представления классификатора, данных выборки:**

Часть данных используется как обучающая выборка, часть - для тестирования. В обучающей выборке считываются все сообщения, подсчитываются повторения слов в классе HAM и SPAM. Для хранения данных используются словари и списки. В словаре хранятся только слова, длиной не менее 4 букв. На 80% от всех файлов строится обучающая выборка, оставшиеся 20% файлов классифицируются. Опираясь на результаты обучающей выборки, рассчитывается вероятность того, что анализируемый файл является сообщением типа SPAM, процент точно принятых решений классификатором (Accuracy), точность (Precision), полнота (Recall). При подсчете и анализе используются встроенные математические функции, для чего импортируются нужные библиотеки.

**4. Написанный программный код с комментариями:**

*# -\*- coding: utf-8 -\*-***import** os  
**import** re  
**import** string  
**import** random  
**from** collections **import** Counter  
**import** math  
**import** decimal  
  
path = **'C:\\Users\\Zver\\PycharmProjects\\spamfilter\\messages\\'**files = os.listdir(path)  
lst = []  
i = 0  
**for** i **in** range(len(files)):  
 lst.append(files[i])  
  
random.shuffle(lst)  
  
train = lst[:int(len(lst)\*80/100)]  
test = lst[int(len(lst)\*80/100):]  
i = 0  
Kolvoham = 0  
Kolvospam = 0  
**for** i **in** range(len(train)):  
 a = train[i].split(**'.'**)[3]  
 **if** a == **'ham'**:  
 Kolvoham = Kolvoham + 1  
 **else**:  
 Kolvospam = Kolvospam + 1  
N = Kolvoham + Kolvospam  
print(**"train\n"**+**"Кол-во СПАМ: "**, Kolvospam)  
print(**"Кол-во НАМ: "**, Kolvoham)  
print(**"Кол-во Всего: "**, N)  
i = 0  
textham = **''**textspam = **''**fham = {}  
fspam = {}  
**'''  
из словарей удаляется пунктуация, цифры, слова, длины меньше трех букв, все слова переводятся в нижний регистр  
'''  
for** file **in** train:  
 a = file.split(**'.'**)[3]  
 **if** a == **'ham'**:  
 f = open(path + file, **'r'**)  
 textham = f.read()  
 f.close()  
 textham = textham.replace(**'!'**,**''**).replace(**'.'**, **''**).replace(**'-'**, **''**).replace(**'@'**, **''**).replace(**'/'**, **''**)  
 textham = textham  
 textham = textham.lower()  
  
 wordham = textham.split()  
 **for** word **in** wordham:  
 word.strip()  
 **if** (len(word) > 3):  
 **if** (word **in** list(fham)):  
 fham[word] += 1  
 **else**:  
 fham[word] = 1  
 **else**:  
 f = open(path + file, **'r'**)  
 textspam = f.read()  
 f.close()  
 textspam = textspam.replace(**'!'**, **''**).replace(**'.'**, **''**).replace(**'-'**, **''**).replace(**'@'**, **''**).replace(**'/'**, **''**)  
 textspam = textspam  
 textspam = textspam.lower()  
 wordsspam = textspam.split()  
  
 **for** word **in** wordsspam:  
 word.strip()  
 **if** (len(word) > 3):  
 **if** (word **in** list(fspam)):  
 fspam[word] += 1  
 **else**:  
 fspam[word] = 1  
  
  
print(**"Мощность словаря НАМ: "**, len(fham))  
Fham = 0  
**for** word **in** list(fham):  
 Fham = Fham + fham[word]  
print(**"Количество слов НАМ: "**, Fham)  
  
print(**"Мощность словаря SPАМ: "**, len(fspam))  
Fspam = 0  
**for** word **in** list(fspam):  
 Fspam = Fspam + fspam[word]  
print(**"Количество слов SPАМ: "**, Fspam)  
  
S = Counter()  
any(map(S.update, (fham, fspam)))  
print(**"Мощность общего словаря: "**, len(S))  
  
**'''  
text - весь текст   
S - словарь   
fspam - словарь спам  
fham - словарь хам  
train - тренирующий текст  
test - тестирующий текст  
'''**predict = []  
true = []  
  
  
**def** testing(i):  
 f = open(path + test[i], **'r'**)  
 testtext = f.read()  
 f.close()  
 Cspam = math.log(Kolvospam / N)  
 Cham = math.log(Kolvoham / N)  
 testtext = re.sub(**'['** + string.punctuation + **']'**, **''**, testtext)  
 testtext = testtext.lower()  
 testtext = re.sub(**'\d'**, **''**, testtext)  
 testtext = testtext.split()  
 **if '.ham' in** test[i]:  
 print(**"HAM"**)  
 print(test[i])  
 **else**:  
 print(**"SPAM"**)  
 print(test[i])  
 **for** word **in** testtext:  
 **if** (len(word) > 3):  
 **if** word **in** fspam:  
 fsp = fspam[word]  
 **else**:  
 fsp = 0  
 **if** word **in** fham:  
 fh = fham[word]  
 **else**:  
 fh = 0  
 Cspam += math.log((fsp + 1) / (len(V) + Fspam))  
 Cham += math.log((fh + 1) / (len(V) + Fham))  
 print(Cspam)  
 print(Cham)  
 decimal.getcontext().prec = 20  
 **if** (Cspam > Cham):  
 Pspam = (1 / (1 + decimal.Decimal(math.exp(Cham - Cspam))))  
 **if** Pspam > 0.8:  
 print(**"SPAM: "**, Pspam)  
 predict.append(0)  
 **else**:  
 print(**"HAM: "**, 1 - Pspam)  
 predict.append(1)  
 **else**:  
 Pham = (1 / (1 + decimal.Decimal(math.exp(Cspam - Cham))))  
 **if** Pham > 0.2:  
 print(**"HAM: "**, Pham)  
 predict.append(1)  
 **else**:  
 print(**"SPAM: "**, 1 - Pham)  
 predict.append(0)  
  
**for** i **in** range(len(test)):  
 **if '.ham' in** test[i]:  
 true.append(1)  
 **else**:  
 true.append(0)  
 testing(i)  
  
i = 0  
P = 0  
**while** (i < len(test)):  
 **if** true[i] == predict[i]:  
 P = P + 1  
 i = i + 1  
i = 0  
TP = 0  
TN = 0  
FP = 0  
FN = 0  
  
i = 0  
**while** (i < len(test)):  
 **if** predict[i] == 1:  
 **if** true[i] == predict[i]:  
 TP = TP + 1  
 **else**:  
 FP = FP + 1  
 **if** predict[i] == 0:  
 **if** true[i] == 0:  
 TN = TN + 1  
 **if** true[i] == 1:  
 FN = FN + 1  
 i = i + 1  
print(P)  
print(**"Истинно-положительных решений: "**, TP)  
print(**"Ложно-положительных решений: "**, FP)  
print(**"Истинно-отрицательных решений: "**, TN)  
print(**"Ложно-отрицательных решений: "**, FN)  
Accuracy = P / len(test)  
print(**"Процент правильно принятых решений классификатором: "**, Accuracy \* 100, **"%"**)

**5. Результаты работы классификатора на тестовой выборке, вывод:**

При запуске программы точность составила порядка 94%, таким образом, можно утверждать, что классификатор достаточно точно отбирает файлы и практически не пропускает нежелательную почту.