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 Десятов А.Г.

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.

Постановка задачи, описание классификатора

Задача фильтра спама – верно установить, что сообщение относится к классу ham или к классу spam. Классификатор определяет, к какому классу относится сообщение с помощью формулы:

$\hat{c}_i = \ln\frac{D^i}{R} + \sum_{n=0}^{N_j} \ln\frac{F^i_{\omega_n^j}+1}{|V|+\sum_{\omega\in V} F^i_\omega}$, где i – номер класса; D^i – количество документов в обучающей выборке, принадлежащих классу c_i; R – общее количество документов в обучающей выборке; j – номер рассматриваемого документа; N_j – количество слов в j-ом документе; $F^i_{\omega_n^j}$ – столько раз n-ое слово j-го документа встречалось в документах класса c_i в обучающей выборке; $|V|$ – количество уникальных слов во всех документах обучающей выборки; $\sum_{\omega\in V} F^i_\omega$ – суммарное количество слов в документах класса c_i в обучающей выборке.

Затем классификатор по $\hat{c}_0$ и $\hat{c}_1$ высчитывает вероятностные оценки того, что сообщение принадлежит классу ham или классу spam. Если вероятностная оценка того, что сообщение относится к классу spam, превышает заданный порог, то классификатор относит данное сообщение к классу spam.

Описание выборки данных для классификации

Выборка данных состоит из 16 545 сообщений, которые принадлежат классу ham, и из 17 157 сообщений, которые принадлежат классу spam. По 80 % сообщений из каждого класса относятся к обучающей выборке, по 10 % – к проверочной выборке и по 10 % – к тестовой выборке.

Описание применяемых структур данных для представления классификатора, данных выборки

Каждое сообщение хранится в отдельном файле, поэтому в программе используется список (list) файлов. С помощью словарей (dict) хранится информация о том, сколько раз слово встретилось в документах определенного класса в обучающей выборке.

Написанный программный код с комментариями

```
import os
import string
import random
import numpy
import pickle


def create_dictionary(dictionary, path, files):
    """Функция создания словаря определенного класса"""
    all_words = 0
    """80 % сообщений для обучающей выборки"""
    doc = int(0.8*len(files))
    """По каждому сообщению"""
    for i in range(doc):
        """Получение текста сообщения"""
        text = read_file(path, files[i])

        """Переделывание текста в список слов"""
        words = text_to_words(text)
        all_words += len(words)

        """Добавление слов в словарь"""
        for wrd in words:
            """Если слово уже есть в словаре"""
            if wrd in dictionary.keys():
                dictionary[wrd] += 1
            else:
                dictionary[wrd] = 1
    """Возвращается суммарное количество слов в словаре"""
    return all_words


def text_to_words(text):
    """Функция переделывания текста в список слов"""
    words_str = ""
    for char in text:
        """Из текста убирются все числа и знаки препинания"""
        if char not in string.punctuation and char not in string.digits:
            words_str += char
    """Текст принимает нижний регистр"""
    words_str = words_str.lower()
    """Получение списка слов из текста"""
    words = words_str.split()
    left = 1
    right = len(words)
    while left < right:
        """Выбрасывание слов из списка, если их длина меньше 3"""
        if len(words[left]) < 3:
```

```
            words.pop(left)
            right -= 1
        else:
            left += 1
    """Вовзвращается список слов"""
    return words


def calculate_c_i(words_mess, dictionary, all_words, doc):
    """Функция подсчета c_i"""
    """Первое слагаемое в формуле"""
    c = numpy.log(doc / all_doc)
    """Знаменатель внутри второго слагаемого в формуле"""
    denominator = V + all_words
    for wrd in words_mess:
        """Числитель внутри второго слагаемого в формуле"""
        numerator = dictionary.get(wrd, 0) + 1
        """dictionary.get(wrd, 0) - возвращает число, сколько раз слово wrd
встретилось в словаре dictionary"""
        c += numpy.log(numerator / denominator)
    return c


def test_spam(message, probability):
    """Функция определяет, является ли сообщение message спамом для порога
probability"""
    """Получение текста сообщения"""
    words_mess = text_to_words(message)

    """Подсчет c0 и c1"""
    c0 = calculate_c_i(words_mess, dict0, all_words0, doc0)
    c1 = calculate_c_i(words_mess, dict1, all_words1, doc1)

    """Дополнительные условия, чтобы избежать исключения"""
    # numpy.exp(-744.1) == 0.0
    # numpy.exp(709.8) == inf
    if -744.0 < c0-c1 < 709.7:
        """Формула soft_max"""
        # p1 = numpy.exp(c1)/(numpy.exp(c0) + numpy.exp(c1))
        p1 = 1 / (1 + numpy.exp(c0 - c1))
    else:
        if c0-c1 >= 709.7:
            p1 = 0.0
        else:
            p1 = 1.0

    if p1 > probability:
        return True
    else:
        return False


def read_file(path, file_name):
    """Функция считывания файла file_name по пути path"""
    path_to_file = os.path.join(path, file_name)
    f = open(path_to_file, 'r', encoding='ANSI')
    text = f.read()
    f.close()
    """Возвращается текст документа"""
    return text
```

```
def read_bin_file(file_name):
    """Функция считывания бинарного файла file name из папки Bin files"""
    path_to_file = os.path.join('Bin_files', file_name)
    f = open(path_to_file, 'rb')
    value = pickle.load(f)
    f.close()
    """Возвращается значение"""
    return value


def write_bin_file(value, file_name):
    """Функция записи value в бинарный файл file_name"""
    path_to_file = os.path.join('Bin_files', file_name)
    f = open(path_to_file, 'wb')
    pickle.dump(value, f)
    f.close()


def test_set(probability, begin0, end0, begin1, end1, print_metrics=False):
    """Функция для Test Set, а также для многократного использования в
Validation Set"""
    tp = tn = fp = fn = 0

    """Анализ сообщений класса HAM"""
    while begin0 < end0:
        if test_spam(read_file(path0, files0[begin0]), probability):
            fp += 1
        else:
            tn += 1
        begin0 += 1

    """Анализ сообщений класса SPAM"""
    while begin1 < end1:
        if test_spam(read_file(path1, files1[begin1]), probability):
            tp += 1
        else:
            fn += 1
        begin1 += 1

    """Вывод дополнительных метрик выполнится, если последний аргумент в функции
True"""
    if print_metrics:
        print('\tprecision \t=', tp / (tp + fp))
        print('\trecall \t\t=',  tp / (tp + fn))
        print('\tf1_score \t=', 2 * tp * tp / (2 * tp * tp + tp * (fn + fp)))
        print('\talf \t\t=', fp / (tn + fp))
        print('\tbet \t\t=', fn / (tp + fn))
        print('\taccuracy \t=', (tp + tn) / (tp + tn + fp + fn))
    """Возвращается accuracy"""
    return (tp + tn) / (tp + tn + fp + fn)


def amount_unique(dict00, dict11):
    """Функция подсчета уникальных слов в двух словарях"""
    unique_key = len(dict00) + len(dict11)
    for key in dict00.keys():
        if key in dict11.keys():
            unique_key -= 1
    return unique_key
```

```
# Train set
print('Train set started!')
"""Пути до файлов с сообщениями ham и spam соответственно"""
path0 = 'Enron/ham'
path1 = 'Enron/spam'

"""Получение списков файлов"""
files0 = os.listdir(path0)
files1 = os.listdir(path1)
"""Перемешивание списков файлов"""
random.shuffle(files0)
random.shuffle(files1)
"""Создание словарей"""
dict0 = {}
dict1 = {}
all_words0 = create_dictionary(dict0, path0, files0)
print('ham dict created! all_words0 =', all_words0)
all_words1 = create_dictionary(dict1, path1, files1)
print('spam dict created! all_words1 =', all_words1)

"""Создание бинарных файлов, чтобы получить результат Train Set без подсчётов"""
# Create bin files "files", "dict" and "all_words"
write_bin_file(files0, 'files0.pickle')
write_bin_file(files1, 'files1.pickle')
write_bin_file(dict0, 'dict0.pickle')
write_bin_file(dict1, 'dict1.pickle')
write_bin_file(all_words0, 'all_words0.pickle')
write_bin_file(all_words1, 'all_words1.pickle')
"""
# Считывание результатов Train Set из бинарных файлов
# Read bin files "files", "dict" and "all_words"
files0 = read_bin_file('files0.pickle')
files1 = read_bin_file('files1.pickle')
dict0 = read_bin_file('dict0.pickle')
dict1 = read_bin_file('dict1.pickle')
all_words0 = read_bin_file('all_words0.pickle')
all_words1 = read_bin_file('all_words1.pickle')
"""

"""80 % сообщений из документов каждого класса"""
doc0 = int(0.8*len(files0))
doc1 = int(0.8*len(files1))
"""Всего сообщений в Train Set"""
all_doc = doc0 + doc1
"""Количество уникальных слов в двух словарях"""
V = amount_unique(dict0, dict1)
print('Train set finished! all unique words =', V)

# Validation set
print('Validation set started!')
"""Начальное значение порога для проверки"""
prob = 0.5
"""Лучшее значение порога"""
best_probability = prob
"""Accuracy для лучшего порога"""
max_acc = 0.0
while prob < 1.0:
    acc = test_set(prob, int(0.9 * len(files0)), len(files0), int(0.9 *
len(files1)), len(files1))
    if acc > max_acc:
```

```
            max_acc = acc
            best_probability = prob
        """Шаг для проверки - 10%: от 50 % до 90 %"""
        prob = numpy.round(prob + 0.1, 1)
print('Validation set finished! The best threshold =',
int(100*best_probability), '%')

# Test set
print('Test set started!')
"""Выполнение Test Set с лучшим порогом"""
test_set(best_probability, int(0.8 * len(files0)), int(0.9 * len(files0)),
int(0.8 * len(files1)),
         int(0.9 * len(files1)), True)
print('Test set finished!')
```

Результаты работы классификатора на тестовой выборке, вывод

Классификатор показал отличные результаты своей работы. Его точность на тестовой выборке составила 98,7 %. Такой фильтр спама годен для анализа текстовых сообщений.