Hw5Ex4

PkGu

12/13/2021

```
import numpy as np
import os


"""
vocab = []
tspm = []
tlgm = []

# Setting up the vocabulary, and words history, of training data.
path_train = "/Users/mac/Downloads/HW5/hw5_nb/train-mails"
name_train = os.listdir(path_train)
name_train.sort()
for fname in name_train:
    dir_train = os.path.join(path_train, fname)
    file = open(dir_train)
    tokens = file.read().split()
    file.close()
    if "spmsg" in fname:
        isspm = 1
    else:
        isspm = 0
    for token in tokens:
        if token.isalpha():
            if token not in vocab:
                vocab.append(token)
            if isspm == 1:
                tspm.append(token)
            else:
                tlgm.append(token)

vocab.sort()
tspm.sort()
tlgm.sort()
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/vocab.npy", np.array(vocab))
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/tspm.npy", np.array(tspm))
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/tlgm.npy", np.array(tlgm))
"""


"""
vocab = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/vocab.npy")
```

```
tspm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/tspm.npy")
tlgm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/tlgm.npy")

num_vcb = len(vocab)
len_spm = len(tspm)
len_lgm = len(tlgm)

# Occurance counter for sorted spam history.
occr_spm = np.zeros(num_vcb)
temp = 0
i_spm = 0
for i_vcb in range(0, num_vcb):
    token = vocab[i_vcb]
    for i_spm in range(temp, len_spm):
        if tspm[i_spm] != token:
            break
        else:
            occr_spm[i_vcb] = occr_spm[i_vcb] + 1
    temp = i_spm

# Occurance counter for sorted legitimized history.
occr_lgm = np.zeros(num_vcb)
temp = 0
i_lgm = 0
for i_vcb in range(0, num_vcb):
    token = vocab[i_vcb]
    for i_lgm in range(temp, len_lgm):
        if tlgm[i_lgm] != token:
            break
        else:
            occr_lgm[i_vcb] = occr_lgm[i_vcb] + 1
    temp = i_lgm

np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/occr_spm.npy", occr_spm)
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/occr_lgm.npy", occr_lgm)
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/len_spm.npy", len_spm)
np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/len_lgm.npy", len_lgm)
"""

"""
vocab = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/vocab.npy")
occr_spm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/occr_spm.npy")
occr_lgm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/occr_lgm.npy")
len_spm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/len_spm.npy")
len_lgm = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/len_lgm.npy")
len_vcb = len(vocab)

# Testing part
path_test = "/Users/mac/Downloads/HW5/hw5_nb/test-mails"
name_test = os.listdir(path_test)
name_test.sort()
isspm_test = np.zeros((len(name_test), 2))
len_test = 0
for i_name in range(0, len(name_test)):
```

```
    fname = name_test[i_name]
    ttest = []
    dir_test = os.path.join(path_test, fname)
    file = open(dir_test)
    tokens = file.read().split()
    file.close()
    if "spmsg" in fname:
        isspm_test[i_name, 1] = 1
    for token in tokens:
        if token.isalpha():
            ttest.append(token)
    len_test = len(ttest)
    bysfct = len_spm / len_lgm
    for i_test in range(0, len_test):
        # Computing spm-likelihood, with Laplace smoothing
        page_sup = len_vcb - 1
        page_inf = 0
        page = 0
        check = 1
        while check:
            if page_sup - page_inf > 2:
                page = round((page_sup + page_inf) / 2)
                if vocab[page] <= ttest[i_test]:
                    page_inf = page
                else:
                    page_sup = page
            else:
                check = 0
                if vocab[page_inf] == ttest[i_test]:
                    page = page_inf
                elif vocab[page_sup] == ttest[i_test]:
                    page = page_sup
                else:
                    page = -1
        if page >= 0:
            llhd_inv_spm = (len_spm + len_vcb) / (occr_spm[page] + 1)
            llhd_inv_lgm = (len_lgm + len_vcb) / (occr_lgm[page] + 1)
            llhd_rt = llhd_inv_lgm / llhd_inv_spm
        else:
            llhd_rt = 1
        # Computing Bayes factor
        bysfct = bysfct * llhd_rt
    if bysfct > 1:
        isspm_test[i_name, 0] = 1

np.save("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/isspm_test.npy", isspm_test)
"""

isspm_test = np.load("/Users/mac/Documents/ML-PSets-icloud/ML-PSet5/NaiveBayes/isspm_test.npy")

# spm -> really spm
isspm_spm = isspm_test[np.where(isspm_test[:, 1] == 1), 0].T
isspm_lgm = isspm_test[np.where(isspm_test[:, 1] == 0), 0].T
```

```
# tr -> correctly predicted, spm -> predicted as spm
tr_spm = np.count_nonzero(isspm_spm)
fs_lgm = len(isspm_spm) - tr_spm
fs_spm = np.count_nonzero(isspm_lgm)
tr_lgm = len(isspm_lgm) - fs_spm

confusion = np.array([[tr_spm, fs_spm], [fs_lgm, tr_lgm]])

precision = tr_spm / (tr_spm + fs_spm)
recall = tr_spm / (tr_spm + fs_lgm)

f_score = 2 / (1 / precision + 1 / recall)

print("Confusion Matrix:\n", confusion)
print("F-Score:\n", f_score)
```