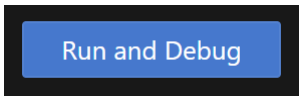
R12725049 徐尚淵 作業三

1. 執行環境：VS code
2. 程式語言：Python (版本 3.11.4)
3. 執行方式：

   直接使用 VS code GUI Run code

4. 作業處理邏輯說明

   1. 讀取需使用的 package

```
import os
import numpy as np
import pandas as pd
from nltk.stem.porter import PorterStemmer
```

   2. 用 Array 與 List 的方式紀錄 Training data 的資訊

```
categories = []

# 開啟檔案
with open('training.txt', 'r') as file:
    # 逐行讀取檔案內容
    for line in file:
        # 切割每一行的數字，去除第一個元素
        category_data = line.strip().split()[1:]

        # 將字串串列加入categories List中
        categories.append(category_data)

#紀錄各類別訓練文件ID
class_docid = np.array(categories)

#所有訓練文件ID
docset = set()
for i in range(class_docid.shape[0]):
    for j in range(class_docid[i].shape[0]):
        docset.add(class_docid[i][j])

train_docid = list(docset)
```

   3. Tokenize Function (同作業一)

```python
def tokenize_text(text):
    # empty List 用以儲存Tokens
    tokens = []

    # empty String 用以儲存單字
    current_token = ""

    separators = [' ', '.', ',', '!', '?', ';', ':', '_', '\"', '(', ')', '\'', '`','@','$','%','&','*','{','/','-','#']

    # 追蹤每個字母
    for char in text:
        # 如果字母是空格或標點符號，並且current string is not empty，則將其添加到tokens List
        if char.isspace() or char.isdigit()  or char in separators:
            if current_token:
                # 添加到 tokens 列表之前檢查长度
                if len(current_token) > 2:
                    tokens.append(current_token)
                current_token = ""
        else:
            # 如果字母不是空格或標點符號，則將其添加到current token
            current_token += char

    # 將最後一個word添加到tokens列表中
    if current_token and len(current_token) > 2 :
        tokens.append(current_token)

    return tokens
```

4. 讀取 Stopwords 並設計前處理 function (同作業一)

```python
# 讀取stopwords
stopwords_file = open("NLTK's list of english stopwords.txt", "r")
stopwords = stopwords_file.read()
stopwords_list = stopwords.splitlines()

def preprocessing(data_path):

    path = data_path
    f = open(path, 'r')
    document_content = f.read()
    f.close()

    processed_texts = []

    # lower casting
    document_content = document_content.lower()
    # Tokenized
    tokenized_content = tokenize_text(document_content)
    # Stopwords removal
    filtered_tokens = [token for token in tokenized_content if token not in stopwords_list]
    # Stemming
    ps=PorterStemmer()
    for t in filtered_tokens:
        processed_texts.append(ps.stem(t))

    return processed_texts
```

5. get_unique function 用於將重複字過濾掉

```python
def get_unique(terms_list):

    list_set = set(terms_list)
    unique_list = (list(list_set))

    return unique_list
```

6. Vocabulary function

首先先將訓練文件 id 以及字彙存成一個 numpy array，再利用 While 迴圈計算字彙出現的次數，最後以 numpy array 的資料型態 return 出去。

```
def Vocabulary(path):
    #對每個trainin data做前處理：用array儲存terms以及相對應的docID
    dir_path = path
    listdir = os.listdir(dir_path)
    term_docid = np.array([[0,0]])

    for i in range(1, len(listdir)+1):
        #以training data製作dictionary
        if str(i) in train_docid:
            data_path = dir_path + str(i) + '.txt'
            term = preprocessing(data_path)
            term = get_unique(term)
            tmp = np.asarray([np.asarray(term), np.repeat(int(i), len(term))]).T
            term_docid = np.concatenate((term_docid, tmp))

    term_docid = term_docid[1:]

    df = pd.DataFrame(term_docid, columns = ['term', 'doc_id'])
    df = df.sort_values(by = ["term", "doc_id"])
    df = df.to_numpy()

    index = 0
    count = 1
    term_df = np.array([[0,0]])

    while(index < len(df)):
        if index+1 < len(df) and df[index][0] == df[index+1][0]:
            count = count + 1
        else:
            term_df = np.concatenate((term_df, np.reshape(([df[index][0], count]), (-1,2))))
            count = 1
        index = index + 1

    term_df = term_df[1:]

    return term_df[:, 0]
```

7. 計算文件數以及將字詞整合的 Function

```
def CountDocs(path):
    return len(train_docid)


def SumTerm(path, class_id):
    data_path = path
    listdir = os.listdir(data_path)

    text = []
    for i in range(1, len(listdir)+1):
        if str(i) in class_docid[class_id-1]:
            dir_path = data_path + str(i) + '.txt'
            term = preprocessing(dir_path)
            text.extend(term)
    return text
```

8. 將文件中 Token 抓出來的 Function

```python
def TokensFromDoc(V, d):
    data_path = "./data/" + str(d) + ".txt"
    tokens = preprocessing(data_path)

    W = []
    for token in tokens:
        if token in V:
            W.append(token)
    return W
```

9. TrainMultinomialNB

利用老師講義的虛擬碼做修改，V 以及 N 就利用前面所寫過的 Function Vocabulary 以及 CountsDoc 來實作取得，分別是一個 array 以及數字，V_Selected 則是透過設計計算的 Chi_Square 的 Function 所得出的 500 個重要字，接著對每個 Class 做迴圈，其中又包含兩個針對 500 個重要字的迴圈計算個字詞屬於某 class_id 文件中的條件機率。

```python
def TrainMultinomialNB(C, path):
    path = path
    V = Vocabulary(path)
    N = CountDocs(path)

    V_Selected = cal_square_test(V, C, path)

    prior = np.zeros(C.shape[0])
    condprob = np.zeros((V_Selected.shape[0], C.shape[0]))

    for class_id in range(1, C.shape[0]+1):
        N_class_id = len(C[class_id-1])
        prior[class_id-1] = N_class_id / N
        text_class_id = SumTerm(path, class_id)

        count_term_sum = 0
        for term in V_Selected:
            count_term = text_class_id.count(term)
            count_term_sum = count_term_sum + count_term

        for term_id, term in enumerate(V_Selected): #term_id starts from 0
            count_term = text_class_id.count(term)
            condprob[term_id][class_id-1] = (count_term + 1)/(count_term_sum + V_Selected.shape[0])

    return V_Selected, prior, condprob
```

10. ApplyMultinomialNB

利用老師講義的虛擬碼做修改，利用 TokensFromDoc 函式獲取文檔的詞彙列表，將每個 class 的先驗機率的對數添加到 score 中，並計算條件機率，最後 return 分數最高的類別，因 array 參數從 0 開始因此最後要加 1

```python
def ApplyMultinomialNB(C, V, prior, condprob, d): #d:doc_id
    W = TokensFromDoc(V, d)

    score = np.zeros(C.shape[0])
    for class_id in range(1, C.shape[0]+1):
        score[class_id-1] = np.log10(prior[class_id-1])

        for term in W:
            term_id = np.where(V == term)[0][0] #term_id starts from 0
            score[class_id-1] = score[class_id-1] + np.log10(condprob[term_id]

    return np.argmax(score)+1
```

11. Feature Selection Function >> 用 Chi Square 來實作

cal_single_chisquare：小的 chi 矩陣，該矩陣包含一個類別的 present 和 absent 數，以及移除掉目前類別的總和，緊接著計算各格的期望值以及 chi-square，累加過後產生最終的 chi-square 值

cal_square_test：對於每個 training data set 的 class 和 Doc，獲取其詞彙列表，並對每個每個詞彙，計算其 chi-square 分數，最後 return 前 500 個 chi-square 分數最高的詞彙作為重要特徵

```python
def cal_single_chisquare(chi_matrix):
    class_num = 13
    doc_num =15
    i = 1
    chi_score = 0
    N_total = class_num * doc_num
    for i in range(13):
        small_chi_matrix = pd.DataFrame(columns=['present', 'absent'], index=[1,2])
        small_chi_matrix = small_chi_matrix.replace(np.nan,0)
        small_chi_matrix.iloc[0] = chi_matrix.iloc[i]
        removed_chi_matrix = chi_matrix.drop([i+1])
        small_chi_matrix.iat[1,0] = removed_chi_matrix['present'].sum()
        small_chi_matrix.iat[1,1] = removed_chi_matrix['absent'].sum()
        E11 =(small_chi_matrix.iat[0,0] + small_chi_matrix.iat[1,0]) * (small_chi_matrix.iat[0,0] + small_chi_matrix.iat[0,1])/N_total
        E01 =(small_chi_matrix.iat[0,1] + small_chi_matrix.iat[1,1]) * (small_chi_matrix.iat[0,0] + small_chi_matrix.iat[0,1])/N_total
        E10 =(small_chi_matrix.iat[0,0] + small_chi_matrix.iat[1,0]) * (small_chi_matrix.iat[1,0] + small_chi_matrix.iat[1,1])/N_total
        E00 =(small_chi_matrix.iat[0,1] + small_chi_matrix.iat[1,1]) * (small_chi_matrix.iat[1,0] + small_chi_matrix.iat[1,1])/N_total
        chi1 = (small_chi_matrix.iat[0,0] - E11)*(small_chi_matrix.iat[0,0] - E11)/E11
        chi2 = (small_chi_matrix.iat[0,1] - E01)*(small_chi_matrix.iat[0,1] - E01)/E01
        chi3 = (small_chi_matrix.iat[1,0] - E10)*(small_chi_matrix.iat[1,0] - E10)/E10
        chi4 = (small_chi_matrix.iat[1,1] - E00)*(small_chi_matrix.iat[1,1] - E00)/E00
        single_chi = chi1 + chi2 + chi3 + chi4
        chi_score = chi_score + single_chi
    chi_score = chi_score / class_num
    return chi_score
```

```python
def cal_square_test(V, C, path):
    total_score = list()
    train_para_list = list()
    for i in range(13):
        single_para_list = list()
        for j in range(15):
            file_code = class_docid[i,j]
            data_path = path + file_code + ".txt"
            term = preprocessing(data_path)
            present_term = get_unique(term)
            single_para_list.append(present_term)
        train_para_list.append(single_para_list)

    for k in range(len(V)):
        chi_matrix = pd.DataFrame(columns=['present', 'absent'], index=[list(range(1,14,1))])
        chi_matrix = chi_matrix.replace(np.nan,0)
        for i in range(13):
            for j in range(0,15):
                if (V[k] in train_para_list[i][j] ):
                    chi_matrix.iat[i,0] = chi_matrix.iat[i,0] + 1
            chi_matrix.iat[i,1] = 15 - chi_matrix.iat[i,0]
        total_score.append(cal_single_chisquare(chi_matrix))
```

12. 主程式部分：讀取檔案、訓練、測試以及輸出

```python
C = class_docid
path = './data/'
listdir = os.listdir(path)
result = []
#training
V, prior, condprob = TrainMultinomialNB(C, path)
#testing
for i in range(1, len(listdir)+1):
    if str(i) not in train_docid:
        data_path = path + str(i) + ".txt"
        class_id = ApplyMultinomialNB(C, V, prior, condprob, i)
        result.append([i, class_id])
df = pd.DataFrame(data = result, columns = ["Id", "Value"])
df.to_csv("new_result.csv", index=False)
print("OK")
```