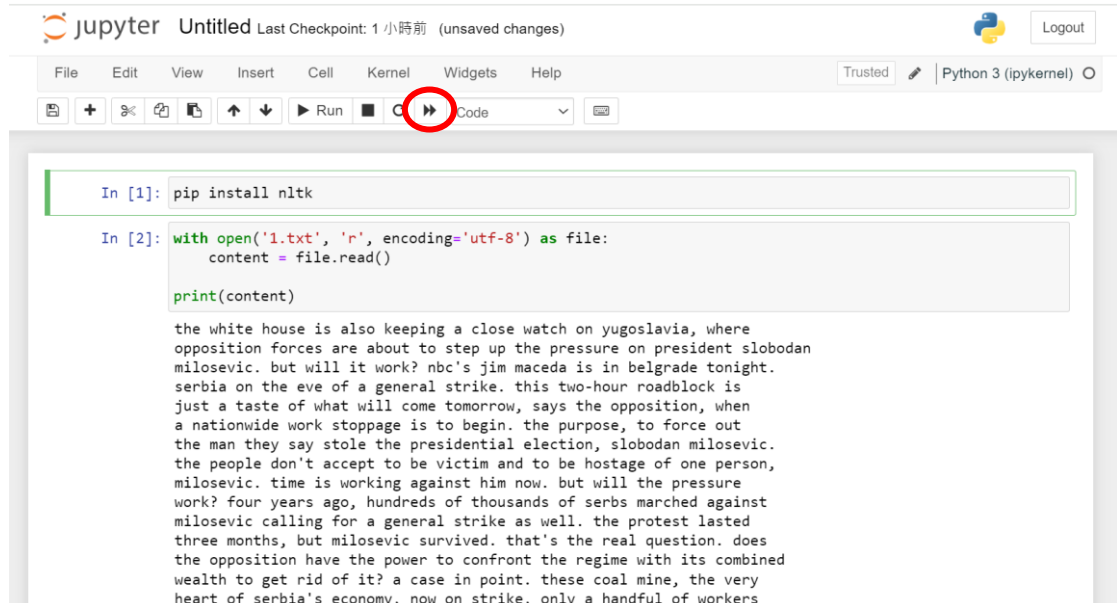


## 文字探勘 HW2

資管三 B11705033 江睿宸

1. 執行環境：Jupyter Notebook
2. 程式語言：Python3



```
In [1]: pip install nltk

In [2]: with open('1.txt', 'r', encoding='utf-8') as file:
        content = file.read()

        print(content)

the white house is also keeping a close watch on yugoslavia, where
opposition forces are about to step up the pressure on president slobodan
milosevic. but will it work? nbc's jim maceda is in belgrade tonight.
serbia on the eve of a general strike. this two-hour roadblock is
just a taste of what will come tomorrow, says the opposition, when
a nationwide work stoppage is to begin. the purpose, to force out
the man they say stole the presidential election, slobodan milosevic.
the people don't accept to be victim and to be hostage of one person,
milosevic. time is working against him now. but will the pressure
work? four years ago, hundreds of thousands of serbs marched against
milosevic calling for a general strike as well. the protest lasted
three months, but milosevic survived. that's the real question. does
the opposition have the power to confront the regime with its combined
wealth to get rid of it? a case in point. these coal mine, the very
heart of serbia's economy, now on strike, only a handful of workers
```

3. 如果沒有 nltk 套件需要先安裝(執行第一個 cell)  
使用 jupyter notebook 執行全部

```
In [2]: import os
import glob
import nltk
from nltk.stem import PorterStemmer
import numpy as np
from collections import defaultdict
import math

nltk.download('punkt')
```

4. Import 需要的套件

```
In [3]: # Tokenize & Calculate frequency
stopwords = ["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", "yours", "your"]

document_frequencies = {} #紀錄單字出現次數
filtered_tokens_list = {} #紀錄所有token
num_files = 1095

for i in range(1, num_files + 1):
    filename = os.path.join('IRTM', f"{i}.txt")
    with open(filename, 'r', encoding='utf-8') as file:
        content = file.read()
        text = content

    # Remove punctuation mark and Lowercasing
    punctuation = ',.!?;:"'()_-'
    punctuation += "'" # except -
    for char in punctuation:
        text = text.replace(char, '')
    tokens = text.lower().split()
```

```

# Stemming
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in tokens]

# Remove stopwords
filtered_tokens = [word for word in tokens if word not in stopwords]
filtered_tokens_list[i] = filtered_tokens

for term in filtered_tokens:
    if term in document_frequencies:
        document_frequencies[term] += 1
    else:
        document_frequencies[term] = 1

```

使用作業一的 `tokenize` 取出 `token`，並計算每個單字的出現次數

```

In [4]: i=1
with open('dictionary.txt', 'w') as f:
    for term in sorted(document_frequencies.keys()):
        f.write(f"{i} {term} {document_frequencies[term]}\n")
        i+=1

```

將結果根據單字升序寫入 `dictionary` 文件中

```

In [5]: tf_list = []
df = defaultdict(int)
for i in range(1, num_files + 1):
    term_frequencies = {}
    for term in filtered_tokens_list[i]:
        if term in term_frequencies:
            term_frequencies[term] += 1
        else:
            term_frequencies[term] = 1

    sum_term = sum(term_frequencies.values())
    tf = {}

    for term, count in term_frequencies.items():
        tf[term] = count / sum_term # 計算 TF
        df[term] += 1

    tf_list.append(tf)

# 計算 IDF
idf = {}
for term, doc_count in df.items():
    idf[term] = math.log(num_files / doc_count)

# 計算 TF-IDF
tf_idf_list = []
for tf in tf_list:
    tf_idf = {}
    for term, tf_value in tf.items():
        tf_idf[term] = tf_value * idf[term]
    tf_idf_list.append(tf_idf)

# 將 TF-IDF 轉換為單位向量
unit_vectors = []
for tf_idf in tf_idf_list:
    magnitude = math.sqrt(sum(value ** 2 for value in tf_idf.values()))
    unit_vector = {term: value / magnitude for term, value in tf_idf.items()} if magnitude > 0 else {}
    unit_vectors.append(unit_vector)

```

根據公式計算 TF-IDF

TF = 計算每個單字在該篇的出現次數 / 該篇單字量

IDF = 根據單字計算總文章數 / 在出現在各幾篇文章中並取  $\log$

相乘後除以長度得單位向量

```

In [6]: terms = sorted(document_frequencies.keys()) # 按詞的字母順序排序
term_to_index = {term: index for index, term in enumerate(terms)} # 單字的編號順序

for i in range(1, num_files + 1):
    filename = os.path.join('Result', f"{i}.txt")
    with open(filename, 'w', encoding='utf-8') as file:
        current_tf_idf = tf_idf_list[i - 1]
        file.write(f"{len(current_tf_idf)}\n") # The number of terms document has

        for term in terms:
            if term in current_tf_idf:
                index = term_to_index[term]
                value = current_tf_idf[term]
                file.write(f"{index} {value:.6f}\n")

```

在 Result 資料夾中根據每個 document 寫入單字的及其 TF-IDF 值

```
In [7]: # 讀取檔案
def load_vector(doc_name):
    vector = {}
    with open(doc_name, 'r', encoding='utf-8') as file:
        next(file) # The number of terms document has
        for line in file:
            index, value = line.split()
            vector[int(index)] = float(value)
    return vector

def cosine(docx, docy):
    vector_x = load_vector(docx)
    vector_y = load_vector(docy)

    # 所有出現的單字集
    all_indices = set(vector_x.keys()).union(set(vector_y.keys()))

    # 統一長度將沒出現的單字填入0
    tf_idf_x = np.array([vector_x.get(term, 0) for term in all_indices])
    tf_idf_y = np.array([vector_y.get(term, 0) for term in all_indices])

    # Calculate the cosine similarity
    dot = np.dot(tf_idf_x, tf_idf_y)
    len_x = np.linalg.norm(tf_idf_x)
    len_y = np.linalg.norm(tf_idf_y)
    if len_x == 0 or len_y == 0:
        return 0.0

    cosine_similarity = dot / (len_x * len_y)
    return cosine_similarity
```

使用 numpy 函數取得內積及兩 vector 長度，並計算 cosine similarity

```
In [8]: x=1
y=2
similarity = cosine(f"output/{x}.txt", f"output/{y}.txt")
print(f"Cosine Similarity: {similarity:.9f}")
```

Cosine Similarity: 0.136289845

在 x y 分別輸入想確認的兩個文件編號，便可得 cosine similarity