# Document for Programming Assignment 3

R06725016 吳亞璇
2017/12/12

---

## 程式語言

- Python 3.5.2

## 執行環境

- Ubuntu 16.04.3 LTS
  - LTS 為 Long Term Support執行方式
- 安裝 Python

```
sudo apt-get install python3.5
```

- 檢查 Python 版本

```
$ python3 --version
Python 3.5.2  # 或任何 3.5.2 以上的版本
```

- 安裝虛擬環境（virtual environment）[註一]

```
sudo apt-get install python3-venv
```

- 建立虛擬環境（virtual environment）

```
python3 -m venv IRTM_venv
```

- 執行虛擬環境

```
$ source IRTM_venv/bin/activate
(IRTM_venv) ~$  # 無論所在路徑為何，(IRTM_venv)必會出現，表示你已經在虛擬環境中
```

- 切換至程式檔所在的目錄

```
~$ cd hw3
(IRTM_venv) ~/hw3$
```

- 此次作業檔案目錄如下

```
hw3
|___ documents          (text collection，共1095個檔案)
|___ data
     |___ training.txt  (training data)
     |--- stopwords.txt
|___ utils.py           (一些共用的function)
|___ vocabs_generation.py (用以產生vocabulary)
|___ hw3.py               (作業主程式)
```

```
|___ vocabs.txt            (經由vocabs_generation.py產生之檔案)
```

- 安裝 Python 套件 — nltk [註二]

```
pip3 install nltk
```

[註一] 使用虛擬環境有許多優點，包括：
- 讓專案有獨立的執行環境。當多人在不同機器上跑同一專案時，也能確保環境的一致性。
- 便於控管套件，避免升級套件時影響到其他專案的執行。

[註二] NLTK（Natural Language Toolkit）
- NLTK 是 Python 的自然語言處理套件，附帶不同程度的預先處理功能，例如：Tokenization
- 此次作業中我只用 NTLK 來實作 Porter's Stemming Algorithm

# 文章分類處理邏輯說明（分為三個部份）

## 產生 vocabulary list

- 和之前的作業一樣，先對 training set 的 195 篇文章做：
  - tokenization
  - stemming
  - remove stop words
  - 得到 terms
- 對 terms 做 **Feature Selection**，選出 500 個 vocabulary 使用於之後的分類模型
  - Feature Selection 的部分實作 **Likelihood Ratio (LLR)**
  - 對每個 term，針對一個 class，計算出 pt, p1, p2
  - 再用 pt, p1, p2 計算出 LH1)、L(H2)，最終算出 -2 log(L(H1) / L(H2))
  - LLR 值越大代表越具有鑑別力，因此留下 LLR 數值較大的 terms
  - 產生 vocabulary list

## Training - 實作 Multinomial NB Classifier

```
for each class:
    for each vocabulary:
        calulate occurence of the vocab in all docs of that class
    total occurence = sum of the occurence of each vocabulary
```

- P(vocab | C) = (occurence of the vocabulary in all docs of class C / total occurence of each vocabulary in all docs of class C)

```
for each class:
    for each vocabulary:
        calculate P(vocab | C)
```

- 得到 class-vocabulary 出現機率的 matrix，可用於 testing 時計算分數

## Testing

- 對每一個文件做和產生 terms (vocabulary) 時相同的處理

```
for each doc:
    put into Multinomial NB Clssifirer and calculate scores of each class
```

```
    final_class = the class with the highest score
```