資訊檢索與文字探勘導論

Programming Assignment 3

資管三 B09705010 吳和謙

1. 執行環境

Jupyter Lab

2. 程式語言 (請標明版本)

Python 3.9.13

3. 執行方式

打開 pa3.ipynb，從頭開始執行每一個 cell，就能夠跑出結果。

我的作業中會用到：

from nltk.stem import PorterStemmer、

import math （用來取 log）。

4. 作業處理邏輯說明

1. 先做 data 的處理，這部分是複製我之前的作業二，去掉一些不需要的步驟。

引入 from nltk.stem import PorterStemmer，讀進 stopwords.txt，並形成 stopWords 的 list。

照順序讀進檔案 1～1095.txt，並分別執行作業一中的以下步驟：

Tokenization:

用內建的 split() 將讀入的字串用空白進行分割，再對每一個分割出來的字串去除 ‘s 、標點符號及數字，存入 tokens 這個 list。

Lowercasing everything:

用內建的 lower() 將每個分割出來的字串變成小寫，存進 lowerWords 這個 list。

Stopword removal:

針對 lowerWords 中的每個字串比對 stopWords 中的字串，若不在 stopWords 中且不是空字串，便存入 filteredList 這個 list。

Stemming using Porter’s algorithm:

將 filteredList 中的每個字串以 ps = PorterStemmer() 進行 stemming 後存入 stemList 這個 list。

最後將每個檔案形成的 stemList 存入 allTokenFile 的 list。

將 allTokenFile 這個二維 list 一維化之後取 set，只留下 unique 的字串，並做 sort 形成升序的字串排列後，變成名為 allTokenSet 的 list（方便進行 index 取值）。

對 allTokenFile 中代表不同 document 的每一列，進行以下步驟：

用 set 留下 unique 的字串，並且 sort 以後存成 curFileSet 的 list。

將 curFileSet 中的字串與 allTokenSet 中的字串進行比較，找出 curFileSet 中字串的 term index，存入 curFileTIndex 這個 list。

對 curFileSet 中的每個字串與 allTokenFile 中該 document 的所有字串進行比較，計算出該 token 的 term frequency 存入 curFileTF 這個 list。

將各個 term index 及 term frequency 以 t_idx: tf 的形式存入名為 cur_tidx_tf 的 dictionary，並在最後將 cur_tidx_tf 存入 allFileTF 這個 list。

2. χ^2 Feature Selection

首先將 “class_1” 到 “class_13” 等十三個字串存入 all_class 的 list 方便未來使用。

讀進檔案 training.txt，並將每一行以空白做分割，去掉此行中的 class_id 之後，以前面提到 all_class 中的每個 class 以 “class_i”: training doc ids 的形式存成 train_data 這個 dictionary。

將所有 training doc ids 存入 all_train_doc 這個 list 中。

將 “all_observed”: 0 和各個 class 以 “class_i”: 0 存入 term_vs_class 這個 dictionary，之後用來儲存各個 term 在 training data 中的 document frequency 以及在各個 class 中的 document frequency。

針對每一個 class 中的每個 training documents，從 allFileTF 中找到文章中的每個 term 並在 all_term_vs_class 將該 class 及 all_observed 加一，因此最後的 all_term_vs_class 會是
{“term_id”: {“all_observed”: n_0, “class_i”: n_i, … }, … } 的形式。

對 all_term_vs_class 中的每個 term 計算 χ^2 的值，並以
“term_id”: χ^2 的形式存入 all_chi2 的 dictionary，其中

$\chi^2 = \sum_{e_t \in \{0,1\}} \sum_{e_c \in C} \frac{\left(N_{e_t e_c} - E_{e_t e_c}\right)^2}{E_{e_t e_c}}$ ， $C = \{1, 2, \ldots, 12, 13\}$

最後，依照 χ^2 將 all_chi2 進行排序後去出前五百個 term 取名為 sorted_chi2_500 作為後續的 features。

3. Multinomial NB Classifier.

Training

首先對所有在 sorted_chi2_500 中的 term 求出在每個 class 中所有 training document 的 term frequency 的總和，並以
“term id”: {“class_i”: n_i, … } 的形式存入 all_tf_class 中。

對於每個 class，求出 prior，並先求出 conditional probability 的分母 sum_tf_curClass，也就是 features 的數量 500 再加上 all_tf_class 每個 class 中的所有 term frequency，再分別求出每個 class 每個 features 的 term frequency + 1 / sum_tf_curClass，以
{“class_i”: {“term id”: n_j, … } … } 的形式存入 condprob。

Testing

引入 math，並將 id, value 寫入要上傳 Kaggle 的 b09705010.csv 檔。

針對每個不在 training data 的其他檔案，對每個 class 求出 score，也就是將 prior 取 log 再加上有出現在該檔案的 features 的 condprob 取 log 的總和。

最後找出 score 最高的 class 作為該檔案的 class，並以
檔案 id, class 的形式寫入該 csv 檔。