# PA3 Report

R04725040 黃柏睿

## Environment:

OSX 10.11, Python 2.7.10

## Report

取出文字的部分 get_stem_voc(doc_id): 利用 PA2 and PA1 來做到 tokenize 跟 stemming ，丟入一篇文章，會回傳一個 dictionary 包含 term 跟 tf

pre-processing
建立一個 dictionary 叫做 trainv_dict，針對每個出現的 term 儲存長度為 14 的陣列（0不用），對應到 13 個 class。
然後把 195 個 training doc 打開(用 get_stem_voc ），針對回傳的 dictionary 出現的 term，在 vlist 填入相對應的 df（如果已經有掃過這個 term，那就取出 14x1 的陣列，然後在對應的的 class 那格 +1，else 沒有這個 term，則建立一個 14格0的陣列，然後把對應的 class 填入 1)

然後針對在 trainv_dict 的每個 term ，跑 LLR 算分數，再把分數存到相對應的 class 跟 term( l_ratio_dict[cid][term] = l_ratio)

接著做 feature selection。
這邊採用的方法是，每個 class 一開始各有 40 次機會，每次輪到一個 class 選擇 feature，選擇方法為該 class LLR 分數最高的取出。class 每次選擇必須用掉一次機會，若選擇的 feature 已經存在 feature_set，則該 class 需要再花一次機會重新找一個 feature(第二高），一直找到不再 feature_set 的 feature，或是用完幾會，然後輪到下一個 class 選擇 feature。
如果每個 class 都用完幾會，但還沒有選到 500 個，則每次重新給予每個 class 5 次機會，直到選出 500 個 feature 為止。
這樣可以確保每個 class 都有一定數量的 term 在 feature set 裡面，並且可以選滿 500 個 feature。

最後是 training 跟 testing
這邊用 score_dict[feature][cid] 來儲存每個 feature 在 training 算出的 NB 分數。
然後再 testing 階段，利用這個 doc 在每個 class 的分數，存在 score_list[cid] 裡面，最後取出 max 在找出他的 index 即為分類所求。