HW1 TF-IDF 實作

tool 使用：使用 pandas、numpy、math

資料讀取

將資料讀取成上方形式，在過程中**發現有個檔案為空白，因此將其移除，在輸出答案時再將其放置到 list 的末端**

將資料傳入自訂的 Function，首先會進行**字串分割**，因此可以將每個 element 存為 dict 裡的 key，以加快速度。而在 TF 及 IDF 的公式部分則做了以下調整：

TF 公式：

把初步 tf 值進行運算：**1 + log (tf)** 得到的結果，

能讓 MAP 評估分數從 0.68 上升至 0.71，因此選擇此公式，

我認為這個公式讓 tf 不會為 0 是提升的關鍵。

IDF 公式：

用 sklearn 套件中的 smooth_idf =True 的公式，

把 idf 值進行運算：**log (1 + (N+1) / (ni+1))** 得到的結果，

與上面的理由相同，我認為他讓 idf 避免除 0 及避免 idf 為 0 是採用的原因

Return：

先將分數的結果進行 L2 正則化，Function 返回值會為所有文章的 TF-IDF 分數及 column 名稱

在評估相似度部分，並未採用 cos_similarity 的公式，原因下方會詳細說明，

VSM 策略

由於 doc 向量的 query 向量的值為少數的 term，因此在文章中不曾出現的機率較高(TF-IDF 值為 0)，導致 norm(doc)接近 0，造成分母為 0 的情況，因此將分母移除，只保留分子的點積，並且實驗發現將 query 向量點積兩次，能得到更好的成果(0.66 → 0.68)

我認為這個方法加大了 query 的影響，使的與 query 相近的資料獲得更高的評分，從而提高了整體表現。

總結與心得

在這次的作業中，一開始還不知道不能使用套件，因此在套件的各種參數上實驗才得到超過 baseline 的成績。寫完才知道要 TF-IDF 部分要自己實作。

因此任務變成：還原套件使用的公式及速度，公式部分在多方查找及比對後很快得到了解決，但是速度方面始終相差甚遠，套件只需要 5 秒，一開始完成的版本卻需要 10 分鐘，後來將 tf 的計算併入 idf 的迴圈中存成陣列才將速度加快成 1 分鐘，不過在這次的作業讓我更熟悉 dic 的運用及好處，也對 TF-IDF 有了更深入的理解。