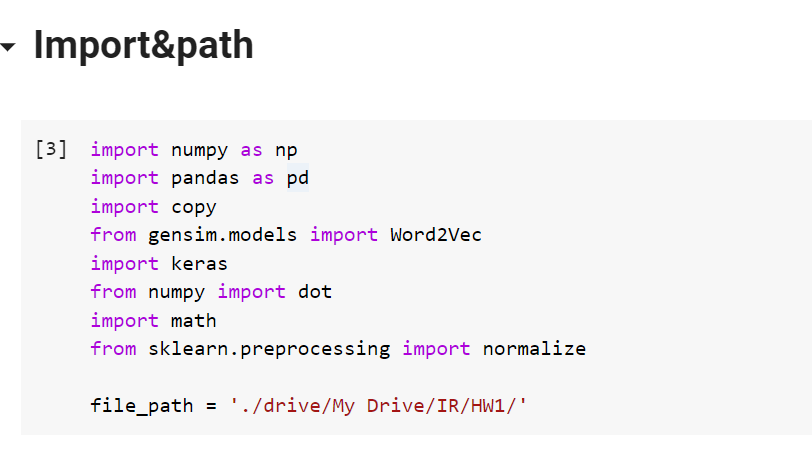
HW1 TF-IDF 實作

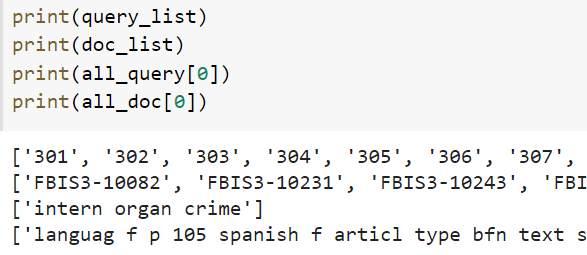
1. **前置作業**

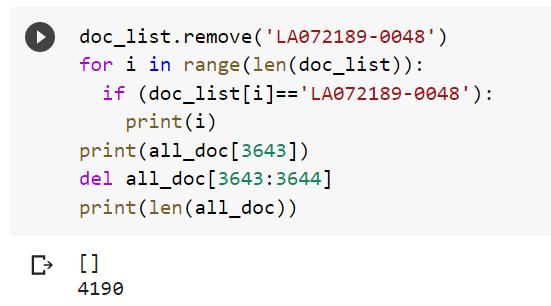




由於使用colab先讓我們能取用google雲端的資料，安裝好套件後import需要的套件及指定檔案路徑

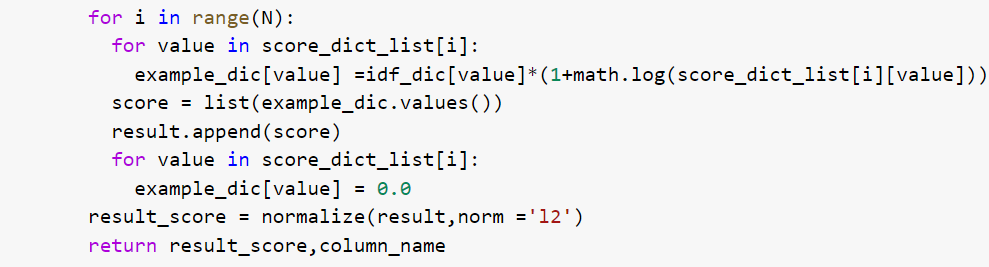
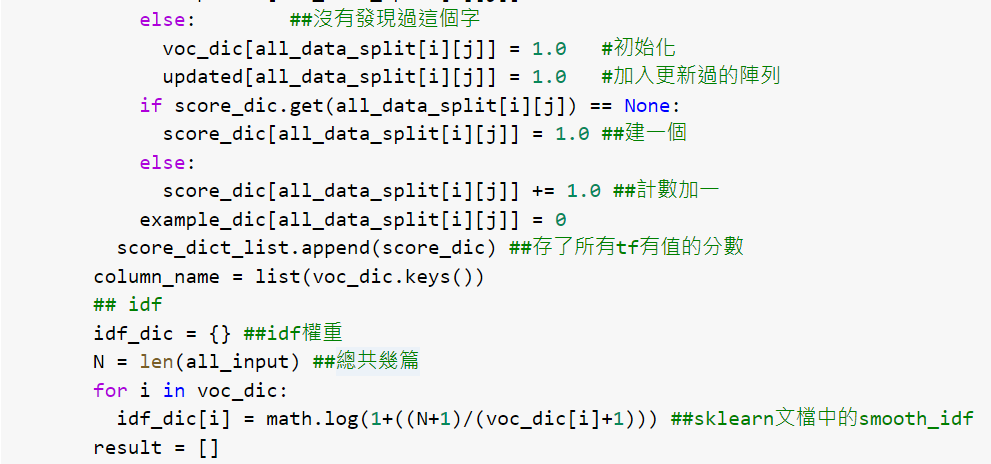
1. **資料讀取**





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

1. **TF-IDF設計**



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，其中各dic的用途如下：

* **example\_dic**：記錄所有key值的dict，方便所有tf進行運算時統一維度，為暫存tf運算結果的dict
* **score\_dic(tf)**：記錄該篇文章中出現過的字及次數，將其append至list得到所有Query及Doc的初步tf值
* **voc\_dic**：記錄所有key值出現在多少篇文章中，單篇最多計算一次
* **updated**：記錄在此篇文章中出現過的key以限制單篇不重複計算
* **idf\_dic**：記錄idf權重的dict

**TF公式**：

用sklearn套件中的sublinear\_tf =True的公式，

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，

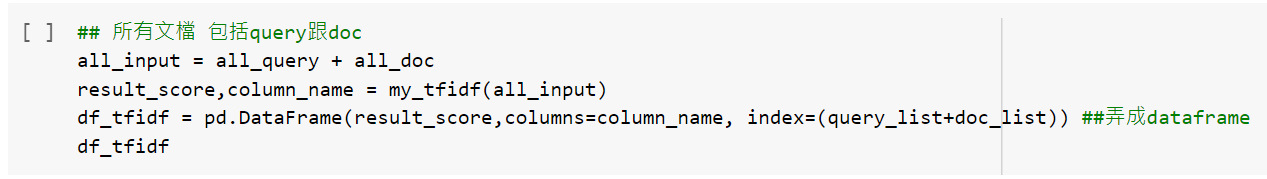
把idf值進行運算：**log ( 1 + ( N+1 ) / ( ni+1 ) )** 得到的結果，

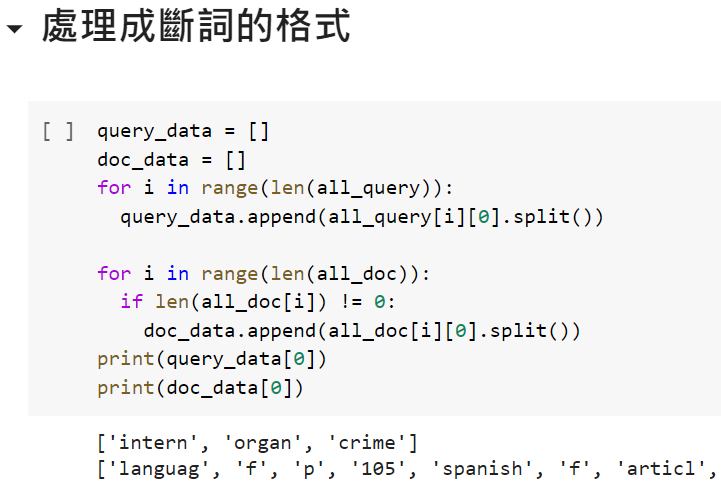
與上面的理由相同，我認為他讓idf避免除0及避免idf為0是採用的原因

**Return：**

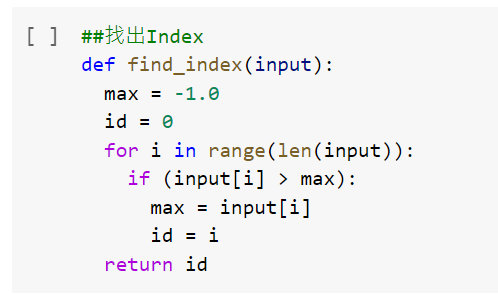
先將分數的結果進行L2正則化，Function返回值會為所有文章的TF-IDF分數及column名稱

1. **實際測試**

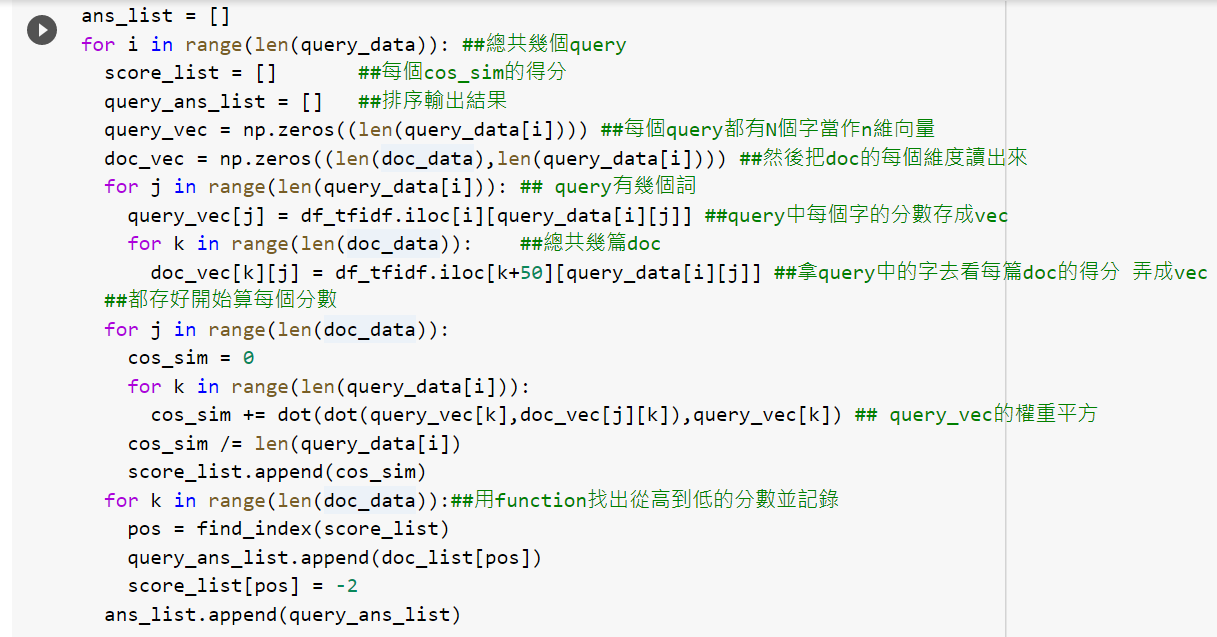




將函式的回傳結果轉成dataframe，並將所有讀取的query、doc進行字串分割得到所有term



尋找index的function

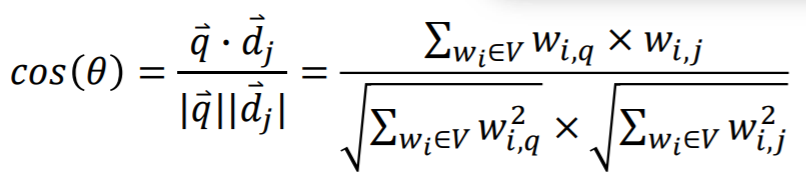


在評估相似度部分，並未採用cos\_similarity的公式，原因下方會詳細說明，

* **評估策略**

由於評估方法為MAP評估，經過查詢後發現高質量的答案對於評估結果較為重要，因此不拿取整個query向量，而是只取要查詢的query中的出現的term當作維度。

**例如**：[‘intern’,’crime’]即為2維，對於每篇doc會去查找doc向量中這兩個的值作為評估標準

由於doc向量的評估值為少數的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。

1. **輸出**



輸出成指定格式，並將錯誤檔案補在最後方。

* **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。