HW1 - Vector Space Model

- **計算 Term Frequency （tf）**

  tf：此 term 在這篇 document 中總共出現了多少次。(local)

  利用二維 Dictionary 的方式將時間複雜度縮減成 O(C)，增加處理效能。

- **計算 Inverse Document Frequency （idf）**

  idf：在所有 document 中此 word 出現的頻率。(global)

  1. **將所有 document 中出現過的 word 存至 all_word_list，利用 set 過濾重複出現的字。**
  2. **再逐一搜尋每篇 document 是否有這個 word，計算頻率儲存至 df_dict。**
  3. **利用 Smooth 公式計算出 idf 值，將其各 + 1 防止分子或分母為 0。**

  ☆ **Smooth 公式中，math 的 log 預設為 e 為底，訓練後的分數為 0.68，改以 10 為底，訓練分數上升至 0.71。**

  ☆ **Smooth 公式是參考 sklearn 中 smooth_idf 參數的 source code，分數有些許的上升：從 0.713 至 0.714**

  ```
  zero divisions: idf(t) = log [ (1 + n) / (1 + df(t)) ] + 1.
  ```

- **計算 tf_idf**

  若 tf_idf 分數越高，則代表此 word 在少量文章出現多次，鑑別性較高，反之分數越低，代表此 word 幾乎在每篇文章都有出現，較沒參考意義。

  ☆ **tf 公式改良是參考 sklearn 中 sublinear_tf 參數的 source code，分數有顯著的上升：從 0.54 至 0.714**

  ```
  Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).
  ```

- **實作 Vector Space Model**

  1. **算出所有 document 和 query 對應的 tf_idf 分數，組合出 vector，若在 tf_idf 的 dictionary 中沒有此 word，則向量為 0。**
  2. **實作 Consine Similarity，將兩個向量做點積後，除以向量之長度。**

  ☆ **技巧：計算 document 的向量時，只取與 query 有關的 term 的 tf_idf 分數當作維度，由於文章中不曾出現的字機率較高，許多 tf_idf 分數為 0，導致分母 norm()趨近於 0，因此我選擇只取與 query 相關性最高的 term。**

- **實作遇到的困難、心得**

  這次作業除了改良公式實作出 VSM 以外，效能也是很重要的因素。最初利用 list 計算 tf、idf 時，跑了一個多小時，最後改用 dictionary 順利將執行時間壓在十秒以內。同時我參考 sklearn 中的 source code 改良公式，順利將分數躍升到 0.71。