WM Programming HW1 Report

R07922134 陳紘豪

1. Describe your VSM

先從 model/ 資料夾底下讀 vocab.all 和 file-list 這兩個檔案，建立資料結構儲存裡頭的 vocabulary 和 file name，並根據其 vocab id 和 file id 將資料放進對應的 index 中，方便之後的存取。再來讀取 inverted-file，並建立一堆資料結構紀錄裡頭的資訊，例如：儲存每篇文章中每個 bigram 或 unigram 的 term frequency、建一個辭典放所有出現過的 term、計算文章的長度、計算每個 term 的 IDF（採用的公式為：$IDF_i = log\frac{N+1}{k}$，其中 N 為 document 數量，k 為 term i 出現在 document 的次數）……等。搜集完這些資訊就可以來做 term frequency 的 okapi normalization 了，其中參數設定是 b = 0.75, k1 = 1.2。

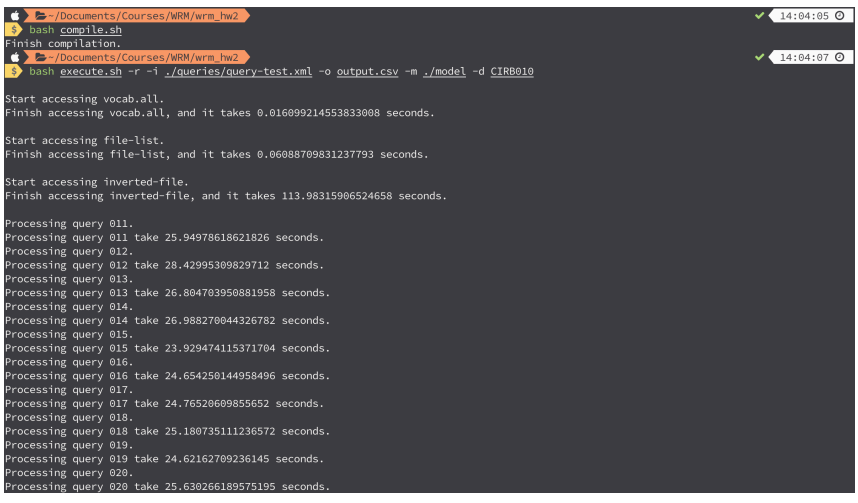
VSM model 所用來表示文章的向量中每一維就是一個曾經在 dataset 中出現的 term，其中的 weight 就是用該 term 的 IDF 乘上文章中該 term 的 normalized TF，至於 query 的讀取就是只讀 XML file 中的 <concepts> 中的詞，並把每一個詞再切成 bigram 以擴增 query，用來表示 query 的向量每一維度和 document 是一樣的（因為在同一個 vector space 中），而其中的 weight 是用 query 中 term 的 normalized TF 來表示，normalize 的方法是用以下公式：$TF_i = \frac{(k_3+1)qtf_i}{k_3+qtf_i}$，參數設定是 k3 = 500。之後就是把 query 和 document 向量做內積算分數，並去排名前 100 名即可。

2. Describe your Rocchio Relevance Feedback

由於不知道 data 中到底哪些是 relevant 哪些是 irrelevant，所以採用 pseudo version 的 Rocchio Relevance Feedback，假設算完分數排完名後的第 1 到第 10 名是 relevant，而排第 101 到第 110 的是 irrelevant，再來就是根據這些 document 來做 query update，使用 RRF 來做 update 會達到 query expansion 的效果，透過 alpha、beta、gamma 這三個參數的設定可以決定新的 query 要偏向舊query、relevant、irrelevant 哪個多一點，而我參數設定為 alpha = 0.5、beta = 0.3、gamma = 0.1。

3. Results of experiments

- Total execution time（with Rocchio Relevance Feedback）：646.92 seconds



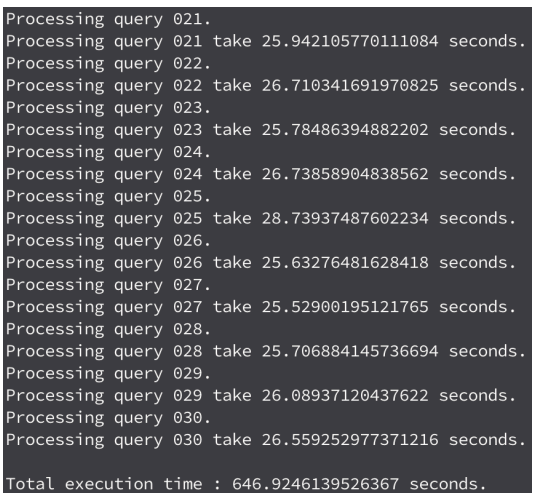
```
~/Documents/Courses/WRM/wrm_hw2                                  14:04:05
$ bash compile.sh
Finish compilation.
~/Documents/Courses/WRM/wrm_hw2                                  14:04:07
$ bash execute.sh -r -i ./queries/query-test.xml -o output.csv -m ./model -d CIRB010

Start accessing vocab.all.
Finish accessing vocab.all, and it takes 0.016099214553833008 seconds.

Start accessing file-list.
Finish accessing file-list, and it takes 0.06088709831237793 seconds.

Start accessing inverted-file.
Finish accessing inverted-file, and it takes 113.98315906524658 seconds.

Processing query 011.
Processing query 011 take 25.94978618621826 seconds.
Processing query 012.
Processing query 012 take 28.42995309829712 seconds.
Processing query 013.
Processing query 013 take 26.804703950881958 seconds.
Processing query 014.
Processing query 014 take 26.988270044326782 seconds.
Processing query 015.
Processing query 015 take 23.929474115371704 seconds.
Processing query 016.
Processing query 016 take 24.654250144958496 seconds.
Processing query 017.
Processing query 017 take 24.76520609855652 seconds.
Processing query 018.
Processing query 018 take 25.180735111236572 seconds.
Processing query 019.
Processing query 019 take 24.62162709236145 seconds.
Processing query 020.
Processing query 020 take 25.630266189575195 seconds.
```



```
Processing query 021.
Processing query 021 take 25.942105770111084 seconds.
Processing query 022.
Processing query 022 take 26.710341691970825 seconds.
Processing query 023.
Processing query 023 take 25.78486394882202 seconds.
Processing query 024.
Processing query 024 take 26.73858904838562 seconds.
Processing query 025.
Processing query 025 take 28.73937487602234 seconds.
Processing query 026.
Processing query 026 take 25.63276481628418 seconds.
Processing query 027.
Processing query 027 take 25.52900195121765 seconds.
Processing query 028.
Processing query 028 take 25.706884145736694 seconds.
Processing query 029.
Processing query 029 take 26.08937120437622 seconds.
Processing query 030.
Processing query 030 take 26.559252977371216 seconds.

Total execution time : 646.9246139526367 seconds.
```

- MAP value under different parameters of VSM

在找比較適合的參數時，因為太多參數數量是變動的，所以一開始就是把 b = 0.75, k1 = 1.2 固定住，還有 Rocchio Relevance Feedback 的那些 alpha、beta、gamma 也固定住，只根據不同的 k3 值來找表現比較好的 model，結果如下：

k3	Public
0	0.30221
500	0.82877
1000	0.82875

最後根據此結果選定 k3 = 500 作為最後的參數。

- With Feedback v.s. Without Feedback

兩種 model 的參數方面都是固定的，唯一不同的就是有無做 Feedback 而已，結果如下：

	Public
With Feedback	0.82877
Without Feedback	0.80535

最後根據此結果決定有 Feedback 的作為 best version。

- Feedback 有無做 Query Expansion

在這個實驗中，我做兩種不同的 Rocchio Relevance Feedback，第一種是原本會做 query expansion 的，也就是把 relevant、irrelevant document 中的 term 加入新的 query 中更新，以獲得新的 query 還有其中每個 term 的 weight，第二種是只對於原本就存在於 query 中的 term 做 weight update 而已，結果如下：

	Public
With Query Expansion	0.82877 (1st)
Without Query Expansion	0.81432

後記：最後我在 kaggle leaderboard 上最後選擇的兩個其實是有 QE 和沒 QE 的這兩種版本，最後在 private 分數上結果如下：

	Private
With Query Expansion	0.70576
Without Query Expansion	0.75074 (10th)

得到的結果是沒有 query expansion 的分數比較高，是個有趣的結果。

4. Discussion

在實作 VSM 的時候發現這個 model 實在是太彈性了，以至於在做參數選擇的時候大多數時候沒有一個更改的方向，不知道到底是哪個參數出了問題，解決辦法是跑去看相關 paper 去看其他人是如何選擇參數，這對我最後 model 的表現還蠻有用的。