
Web Retrieval and Mining spring 2013

Assignment 3 (Programming HW 1)
Ranking (due on 4/12)

資管四 徐承志 B98705034 · NTUIM · 2013/04/11



目錄



how to execute your program	2
execution time with your hardware spec	3
your program design & procedure	4
Code Framework	10
advantage of your program	12
discussions	12
Resource & Reference	14

1. HOW TO EXECUTE YOUR PROGRAM

Execution Environment :

Mac OSX、Linux with Ruby Programming Language. If you do not have Ruby setup, please install ruby first.

	Version
OS	Ubuntu 12.04 64 bit
Ruby	2.0.0

```
[~] $ ruby -v
ruby 2.0.0p0 (2013-02-24 revision 39474) [x86_64-darwin12.3.0]
```

Getting Start :

終端機指令	說明
\$ which bash => /bin/bash	查看bash 位置
\$ chmod +x compile.sh	調整權限
\$ chmod +x execute.sh	
\$./compile.sh	compile 助教給的Text processing toolkit (Special thanks to Shao Hang Kao (R97922009))
\$./execute.sh [-r] -i tmp2/queries/ query-5.xml -o ans -m tmp2/model-files -d tmp2/CIRB010	執行程式

```
$ ./execute.sh -r -i tmp2/queries/query-5.xml -o ans -m tmp2/model-files -d tmp2/CIRB010
```

2. EXECUTION TIME WITH YOUR HARDWARE SPEC

機型名稱	Linux 3.5.0-27-generic
處理器名稱	Intel Core2 Quad CPU Q9400
處理器速度	2.66 GHz
總核心數目	4
記憶體	7.8 GB
Execution Time	10 min for Query-5



3. YOUR PROGRAM DESIGN & PROCEDURE

Parse Query XML topic

因為一開始的 input file 是多個 queries 組成的 XML 檔案，所以在 input file 剛進來的時候我進先將它作處理，然後測試過後截取的 XML 標籤 (title、question、narrative、concept) 不論是全取或是去掉 title 不去斷字成 query 的 term 實在都不太影響最後的 MAP 結果，所以我把一整份 queries file 切成小份的單一 query 的形式去執行。

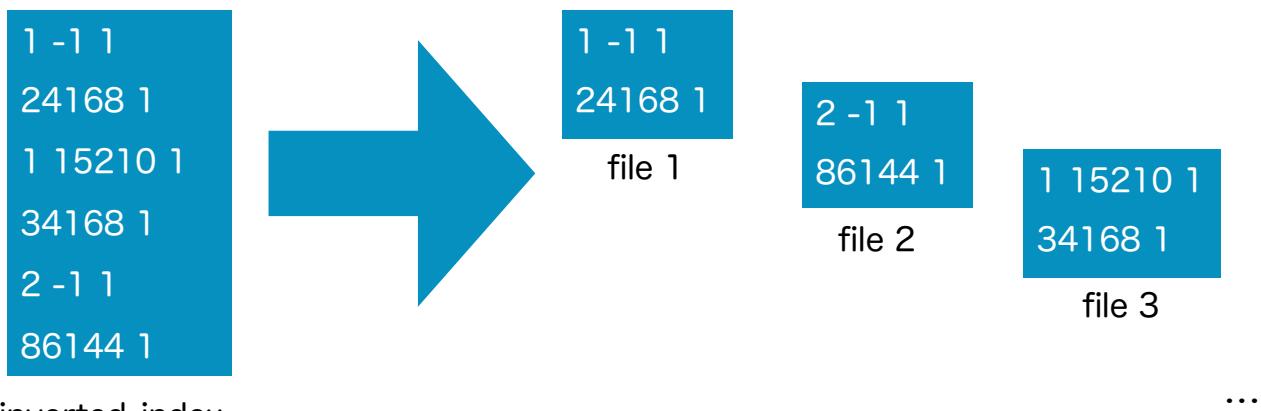
下圖為 parse 全部 query 的執行過程。以及切割後單一的 XML file 檔案。

```
michaelhsu@TutorNTU: ~/Programming assignment1
Parse query2...
Loading vocabulary file...Complete!
Parse query3...
Loading vocabulary file...Complete!
Parse query4...
Loading vocabulary file...Complete!
Parse query5...
Loading vocabulary file...Complete!
Parse query6...
Loading vocabulary file...Complete!
Parse query7...
Loading vocabulary file...Complete!
Parse query8...
Loading vocabulary file...Complete!
Parse query9...
Loading vocabulary file...Complete!
Parse query10...
Loading vocabulary file...Complete!
Parse query11...
Loading vocabulary file...Complete!
Parse query12...
Loading vocabulary file...Complete!
Parse query13...
Loading vocabulary file...Complete!
Parse query14...
Loading vocabulary file...Complete!
Parse query15...
Loading vocabulary file...Complete!
Parse query16...
Loading vocabulary file...Complete!
Parse query17...
Loading vocabulary file...Complete!
Parse query18...
Loading vocabulary file...Complete!
Parse query19...
Loading vocabulary file...Complete!
Parse query20...
Loading vocabulary file...Complete!
Parse query21...
Loading vocabulary file...Complete!
Parse query22...
```

```
query1.xml      *  main.rb      *  method.rb      *  query-30.xml     *  query-5.xml      *  compile.sh
1  <topic>
2  <number>CIRB010TopicZH001</number>
3  <title>集會遊行法與言論自由</title>
4  <question>
5  查詢集會遊行法中有關主張共產主義或分裂國土規定之修正與討論。
6  </question>
7  <narrative>
8  相關文件內容應敘述集會遊行法原本對主張共產主義或分裂國土之限制，其是否符合憲法中對言論自由等基本人權的保障，大法官對此議題的相關解釋，等
9  </narrative>
10 <concepts>
11 集會遊行法、集會遊行、集遊法、憲法、言論自由、保障、共產主義、分裂國土、大法官會議、立法、修正條文。
12 </concepts>
13 </topic>
```

資料結構：

為求縮減 Debug 的時間，處理上我先將助教提供 model-files 底下的 inverted-index 作資料結構轉換，將一大份完整的 inverted-index file 檔案做切割，切成小份小份的 file 檔案，命名以 vocab_id 為檔名。



inverted-index

A screenshot of a terminal window titled "michaelhsu@TutorNTU: ~/Programming assignment". The command entered is:

```
"10386_8031"  
"10386_8033"  
"10386_8037"  
"10386_8046"  
"10386_8049"  
"10386_8053"  
"10386_8055"  
"10386_8058"  
"10386_8059"  
"10386_8065"  
"10386_8070"  
"10386_8072"  
"10386_8076"  
"10386_8078"  
"10386_8079"  
"10386_8081"  
"10386_8083"  
"10386_8084"  
"10386_8092"  
"10386_8095"  
"10386_8096"  
"10386_8097"  
"10386_8100"  
"10386_8102"  
"10386_8103"  
"10386_8105"  
"10386_8108"  
"10386_8112"  
"10386_8113"  
"10386_8114"  
"10386_8116"  
"10386_8118"  
"10386_8119"  
"10386_8121"  
"10386_8122"
```

上圖正在處理檔案資料結構的切割轉換，需時大概 10 分鐘。

VSM model Similarity Scoring :

採用的演算法是 Fig. 7.1: a fast way to calculate cosine scores on VSM with the assistance of inverted file.

```
FASTCOSINESCORE( $q$ )
1 float Scores[N] = 0
2 for each  $d$ 
3 do Initialize Length[d] to the length of doc  $d$ 
4 for each query term  $t$ 
5 do calculate  $w_{t,q}$  and fetch postings list for  $t$ 
6 for each pair( $d, tf_{t,d}$ ) in postings list
7 do add  $wf_{t,d}$  to Scores[d]
8 Read the array Length[d]
9 for each  $d$ 
10 do Divide Scores[d] by Length[d]
11 return Top K components of Scores[]
```

► Figure 7.1 A faster algorithm for vector space scores.

在 query 的 weight 上, 我忽略 query 的 tf, 直接把 $w_{t,q}$ 視作 1, 然後在算每份 Document file 與 Query 的 similarity Scores 時, Document 的 tf 先做 term normalize, 使用 Okapi/BM25 TF Normalization 的方式

$$\text{Norm TF} = \text{Raw TF} / (1 - b + b * \text{doclen} / \text{avgdoclen})$$

其中把 doclen 以 Document file 的 byte size 去計算。

接著用 normalize 後的 tf 去乘以該份 Document 的 idf , 得以得到這份 Document 每個 Vector 的 weight, 然後使用 Dot product similarity 去計算分數。

$$\text{Dot product similarity: } sim(\bar{Q}, \bar{D}_i) = \sum_{j=1}^N w_{qj} * w_{ij}$$

不使用 Cosine similarity (= normalized dot product) 去算 Scores 的原因是，一開始有嘗試這樣做，但是後來發現 Ans-5 的結果 MAP (取 Top 30, MAP = 0.34) 表現不好，才改以先將 tf 先做 normalize，再做 Dot product similarity。

決定 K 值的嘗試：

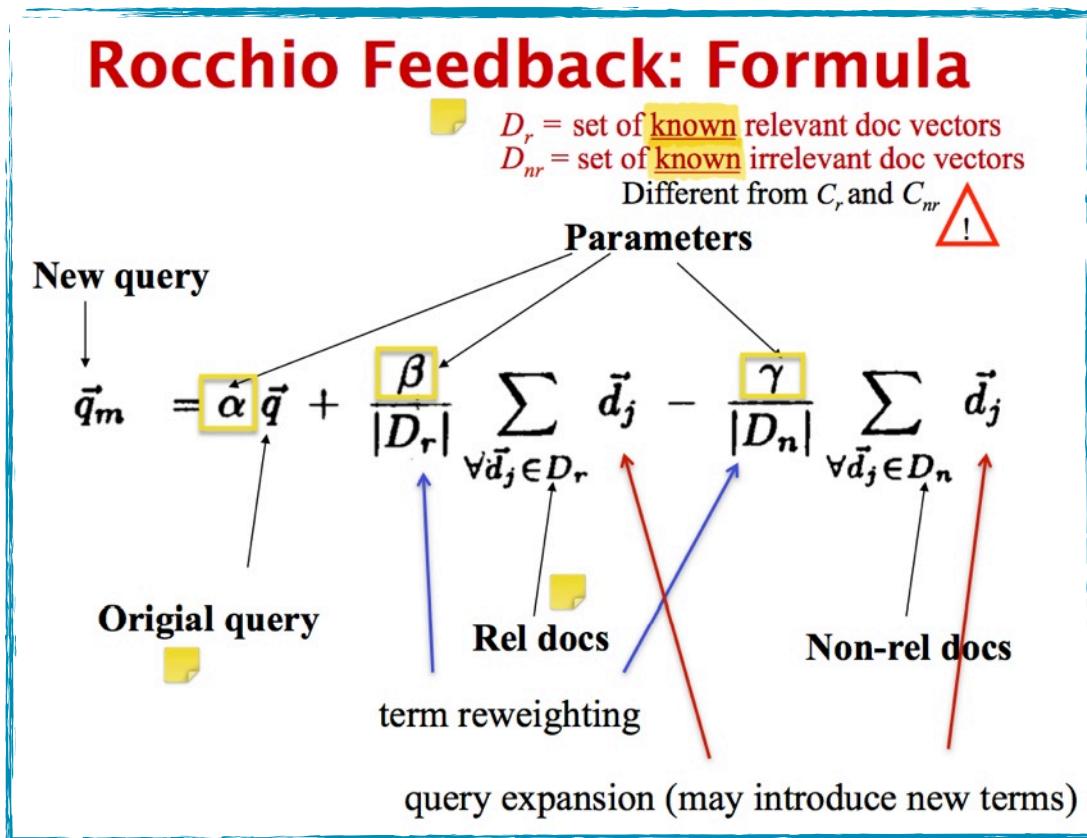
一開始在我的 model 下考慮unigram, bigram 兩者權重相同來測試，因為不太能找出一 threshold 分數，所以取 top K relevant 來作測試

嘗試	K	MAP
Cosine similarity (0.5)unigram (0.5)bigram	30	0.43
	40	0.33
先將 tf 作normalize 再算 Dot similarity (0.5)unigram (0.5)bigram	20	0.48
	30	0.51
	40	0.52
	50	0.55
	80	0.57
	90	0.58
	100	0.58
先將 tf 作normalize 再算 Dot similarity (1)unigram (0)bigram		表現很差
先將 tf 作normalize 再算 Dot similarity (0)unigram (1)bigram	40	0.52
	100	0.57
先將 tf 作normalize 再算 Dot similarity (0.2)unigram (0.8)bigram	40	0.53
	50	0.55
	80	0.57
	100	0.58
先將 tf 作normalize 再算 Dot similarity	40	0.53

嘗試	K	MAP
(0.3)unigram (0.7)bigram	80	0.57
先將 tf 作normalize 再算 Dot similarity	30	0.5
(0.3)unigram (0.7)bigram	40	0.53
	80	0.57

最後採用只考慮 bigram only, top 100 as relevant.

Rocchio relevance feedback :



我的實作方法為 原先的結果有一個 Scores 的 Array , 存放原本每份 Document 對應結果的 Score, 然後依照取出的前 R 份當作 Relevant docs, 作為feedback , 做法為透過助教提供 Text processing toolkit 中的 Create-ngram、merge-ngram 來 Merge 前 R 份的 Relevant Docs , 然後再以 alpha 、 Beta 直來作權重分配, 嘗試多次 Feedback N 個 loop次, 取出最後結果的 Top 100 當作 ans-feedback 輸出的結果。

loop N	q alpha	TOP R	MAP
1	0.5	2	0.601
	0.6	2	0.594
	0.7	2	0.58
	0.8	2	0.571
1	0.5	3	0.56
	0.5	4	0.608
	0.5	5	0.53
2	0.5	4	0.591
3	0.5	4	0.612
4	0.5	4	0.613
5	0.5	4	0.602

最後採用 $\alpha = 0.5$, $top R = 4$, do $N = 4$ feedback loop。

```

[-/Programming assignment1] $ ./ex
{:r=>true, :i=>"../tmp2/queries/que
The Inverted-index Structure has c
Scoring query1_ngram.all...
Scoring query1_ngram.all with Rocchio... in loop 1 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query1_ngram.all with Rocchio... in loop 2 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query1_ngram.all with Rocchio... in loop 3 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query1_ngram.all with Rocchio... in loop 4 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query2_ngram.all...
Scoring query2_ngram.all with Rocchio... in loop 1 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query2_ngram.all with Rocchio... in loop 2 ...
Loading vocabulary file...Complete!
Loading vocabulary file...Complete!
Scoring query2_ngram.all with Rocchio... in loop 3 ...

```

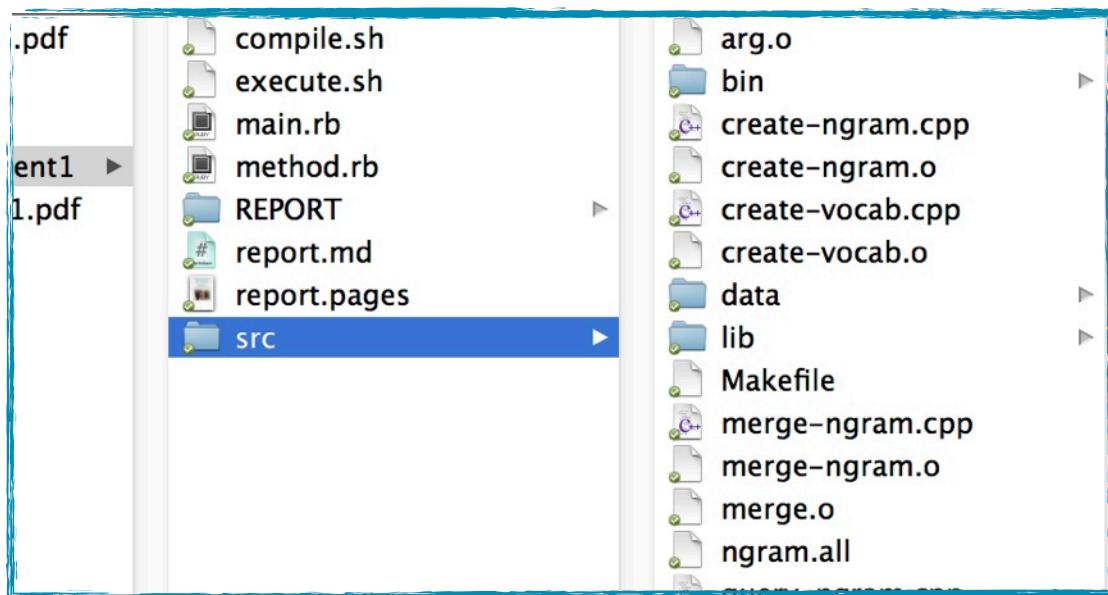
Scoring

feedback loop 1

feedback loop 2

4. CODE FRAMEWORK

下圖為 整份程式的架構。



main.rb	只要的程式執行順序
shell script 主要執行	include 所有 method.rb 去執行。

method.rb	定義了五個 Function
Convert_InvertedIndexFile()	將原本的 inverted-index 作資料結構轉換。
FastCosineScore(query)	VSM 主要的model 建制，用來計算 Scores，回傳一個 Document 對應 Score 的 Hash。input merged ngram query lines array, output scores Hash
ParseQuery(n)	query-5、query-30，將 input 進來的 XML 檔做 parse 存成小份小份的 query file。input query-5 or query-30 file, n, output ngram.all (file in disk)
PickTopK (scores, topK)	Rank Hash 的 value, 後回 top K 的 document id

method.rb	定義了五個 Function
Rocchio(queryi, scores, scores_topk, topK)	Rocchio Relevance Feedback (pseudo version), input relevant array, fetch topK, output top K relevant array

下圖為 method.rb 的程式架構。

```

1  ### structure convert, input path of model-files, output X
2  def Convert_InvertedIndexFile() ...
46 end
47 ###
48
49 ### Score similarity, input merged ngram query lines array, output
50 def FastCosineScore(queryi) ...
137 end
138
139 ### query parser, input query-5 or query-30 file, n, output ngram
140 def ParseQuery(n) ...
168 end
169
170 def PickTopK (scores, topK) ...
173 end
174
175 ### Rocchio Relevance Feedback (pseudo version), input relevant a
176 def Rocchio(queryi, scores, scores_topk, topK) ...
187 end

```

4. ADVANTAGE OF YOUR PROGRAM

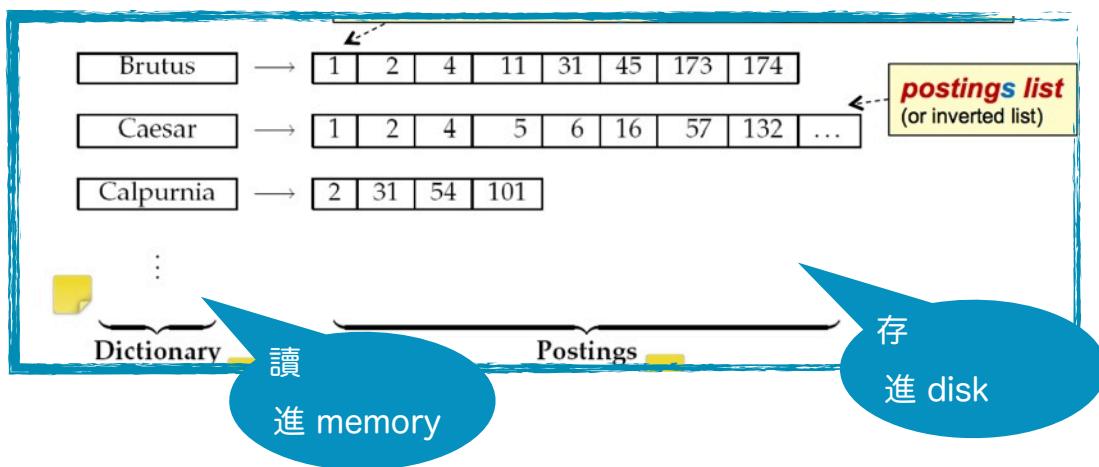
因為挑選了 Script language Ruby , 所以在實作的時候可以節省很多時間，然後在處理資料結構上，我認為先將大份的檔案作拆解才能縮減你建制 model 需要 Debug 的時間，才不用每次重新 run 你個程式都花個十分鐘再讀取 file 。

5. DISCUSSIONS

在做這次的程式作業令我困難點有以下幾個：

1. 資料結構的考慮

一開始我是用我的筆電跑程式，關於 inverted-index 這份檔案，雖然在處理上很方便，但是每當我程式要讀進整份檔案 memory 就會爆掉，所以我後來的處理方式是把 vocab_id 讀進記憶體，然後把每一個 vocab_id_1 vocab_id_2 組成的 term 所對應的 postings-list (數個 file_id) 放進disk 存起來 (新增 file) ，雖然如此一來可以解決記憶體的不足情況，但是這樣新增的 files 檔案容量達到 5G以上的空間。但是很不幸的我的 disk 也只剩1G的空間，所以也沒辦法，只好換台電腦繼續做。



2. score 的演算法

在考慮 unigram 以及 bigram 上，我認為中文不像英文，中文反而是兩個字組成的單詞會比較有代表性，後來我實驗出來的結果也是如此，不論 unigram 參與的權重多少，好像都不太影響 MAP 的分數。

上完課我原本直覺 cosine similarity 的算法應該是比較好的，但是後來實驗結果也非如此，所以後來才採用先 tf normalize , 再算 dot similarity 。

3. feedback 以及各種參數的調整

說到 tune 參數，我發現除了一開始的資料結構處理，這裡是最花時間的，因為真的有非常多的參數可以調整，像是取 top K 多少 as relevant 呢？feedback 要取前幾個呢？要做幾次 feedback 呢？merge 後的個別權重要多少？我只不過做了少不份的嘗試就花了將近一天的時間，但是結果時好時壞。

4. 實作語言與工作站問題

圖破了重重困境，我終於完成所有要求，然後寫完shell script檔了，在本機端可以跑完沒問題。但是助教說要在R217上跑過才算，後來我就丟上去跑跑看，但是碰到的問題多半是權限的問題，譬如我會新增一個資料夾來暫存我轉換過後的資料格式，但是新增資料夾的時候就沒有權限，後來chmod 777才可以。另一個問題我一直沒辦法解決就是 stack too deep (SystemStackError)，我目前搜尋發現可能是ruby 版本的問題，但是我也沒辦法去更改工作站上的版本。

6. RESOURCE & REFERENCE

1. ruby http://www.ruby-lang.org/zh_TW/downloads/
2. ruby array <http://www.ruby-doc.org/core-1.9.2/Array.html>
3. ruby hash <http://www.ruby-doc.org/core-1.9.2/Hash.html>
4. [Web Retrieval and Mining spring 2013](#)
5. IIR book <http://nlp.stanford.edu/IR-book/>