Web Retrieval and Mining spring 2013
Programming HW1

B98705024 資管四 陳昱安

Specify the problem

這次程式主要在建立 VSM 的系統，並搭配 Rocchio Feedback 功能。透過不同的 query 去操作，從龐大的資料文章中選取與 query 最相關的文章。(這次的資料文章有 97445 筆 XML 資料，以及助教提供存有文章列表的 file-list、存有所有 vocabularies 的 vocab.all 和已經建立好 inverted-index 的 file。還能透過 Mean Average Precision 來檢測系統效能，方便改善)

Construct model

一開始先將 inverted-index 讀進系統，存入 memory 中，以便未來方便使用，這邊使用 object 的形式，去存取每個 document 中的每個 term 的向量。

建立 VSM 系統中，主要使用基本的 TF * IDF，不過有對 TF 做 normalization。Normalize 方面是採用 Okapi/BM25 的方式

$$TF(t,d) = (k+1)\, f(t,d)/(f(t,d)+k(1-b+b*doclen/avgdoclen))$$

其中參數 k 是 2，b 是 0.75 為 Default，後面會有一些不同參數的嘗試。

IDF 就如一般一樣

$$IDF(t) = \log(n/k)$$

n – total number of docs
k – # docs with term t (doc freq)

在選取 Term 時，我目前只先採取單字處理。在算完分數後，選取相關文章，這邊我做了幾種嘗試。選出分數最高的前 30 篇文章、前 35 篇文章、前 40 篇文章、前 50 篇文章，以及先找出分數最高的文章，然後將每篇文章的分數除以這個分數最高的文章最一個正規化處理，然後選取大於 0.5 的當做相關文章。

(Score / Max_Score > 0.5)

接著，若有選擇 Feedback 的話，會將分數最高的前 k 篇文章取出 (這邊我

做的 5 和 10)，將這些文章的 term 重新組成一個 new query，在去重新算一次分數，不過這邊的比重 old query : new query = 9 : 1。(Alpha = 0.9 Beta = 0.1 後面會有其他參數的嘗試)

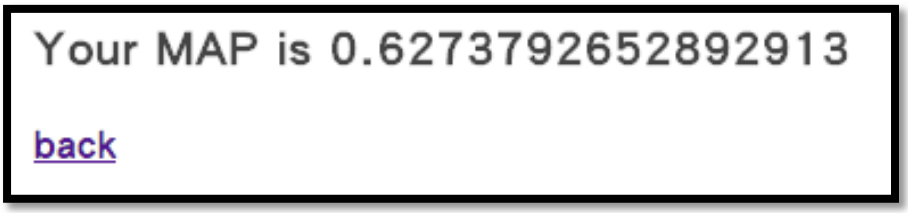
最後分數= 0.9*query 算出來的分數 + 0.1 文章組成的 query 算出來的分數

Results of Experiments

以下是各個不同參數時，用 query-5.xml 算出來的 MAP

###

單字，沒有 feedback，用正規化大於 0.5 取相關文章：0.627



單字，沒有 feedback，Okapi/BM25 中 k=1.2 b=0.75，用正規化大於 0.5 取相關文章：0.60

單字，沒有 feedback，Okapi/BM25 中 k=2 b=0.85，用正規化大於 0.5 取相關文章：0.629

單字，沒有 feedback，Okapi/BM25 中 k=1 b=0.75，用正規化大於 0.5 取相關文章：0.582

單字，沒有 feedback，取分數最高的前 30 篇當做相關文章：0.6016

單字，沒有 feedback，取分數最高的前 35 篇當做相關文章：0.61

單字，沒有 feedback，取分數最高的前 40 篇當做相關文章：0.61

單字，沒有 feedback，取分數最高的前 50 篇當做相關文章：0.607

###

單字，有 feedback，feedback 取前 5 篇分數最高的文章，用正規化大於 0.5 取相關文章：0.6131

單字，有 feedback，feedback 取前 5 篇分數最高的文章，Beta=0.75，用正規化大於 0.5 取相關文章：0.582

單字，有 feedback，feedback 取前 5 篇分數最高的文章，重複做三次 feedback，用正規化大於 0.5 取相關文章：0.1648

###

雙字，沒有 feedback，用正規化大於 0.5 取相關文章：0.5868

Discussion

從 MAP 看出，當在做判斷相關文章時，用 TOP K，取分數最高的前幾名，效果會比較差，因為有可能都很不相關的文章被取出來，即使分數很低，採取正規化當判斷更加適當。

這次系統花了很多時間在做前置作業，反而在建置 VSM 的計算相對簡單。這樣表示其實前置作業(建立 inverted-index)在 Data Mining 中是佔有很重要的地位。

從 Okapi/BM25 的 normalization 參數測試，發現這些參數都些微影響結果，可能是調整的不夠顯著，反倒是有 Feedback 的分數都會下降，我想是因為一開始的結果不夠好，取出的文章並非相關性很高，導致 Feedback 的 Term Vector 也與真正相關的差距甚遠。

這次沒有妥善處理 Stop Word 的問題，雖然在 IDF 中可以消除 Stop Word 影響文章的 Rank，但是有可能嚴重拖慢系統執行效率，因為 Stop Word 容易在每篇文章都有出現，故針對每篇文章處理時，就會一直處理到沒用的 Stop Word，下次可以考慮先將 Stop Word 從 inverted-index 中去除，可以增加更多效能。

這次的作業有個深刻的體會，就是跟所有資源在競爭，為了在工作站上執行，硬碟空間不能使用太多，但不存 file，記憶體空間又會不夠，又要講求速度，又要準確度，算是體會到在有限資源做最大利用。