B04902060 周良冠

Web Retrieval and Mining spring 2019

Programming HW 1 Report

一、實驗架構

1. Data 預處理：讀入 inverted-file，分別建立 terms 和 documents 的字典。
2. Query 預處理：讀入 xml 檔裡 concepts 的詞彙，查找當中合法的 unigram terms 和 bigram terms，得到代表 query 的向量。其中，向量元素的值預設為 1。
3. VSM Model：讓每個 documents 跟 query 計算分數，取前 100 名作為預測結果。其中實驗包括兩個不同的算分方式：單純 TF/IDF 以及 Okapi。
4. Rocchio Feedback：經過 VSM 得到預測結果後，將前後幾名 documents 的向量與 query 的向量相加，重新做 VSM。

二、實驗結果與分析

1. VSM Model

首先使用沒有標準化的 TF/IDF，得到的分數非常差。套用了課堂介紹的 Okapi，並且根據 Okapi 作者建議，k1=[1, 2]，b=0.75, ka=[0, 1000]，分數就能直接超越 strong baseline。我試驗了五組不同的參數，發現當 k1=1.5 時分數最高，而 ka 則對結果沒什麼影響。

TF/IDF：無 normalized 過。TF(t, d) = c(t, d), IDF(t) = log(n/k)

Okapi(k1, b, ka)：對應原論文的參數表示方法。

method	Public score
TF/IDF	0.48949
Okapi(1, 0.75, 500)	0.81860
Okapi(1.5, 0.75, 500)	**0.82237**
Okapi(2, 0.75, 500)	0.82077
Okapi(1.5, 0.75, 250)	0.82237
Okapi(1.5, 0.75, 750)	0.82237

2. Rocchio Feedback

每次 feedback 會挑出 VSM 分數最高的 kr 個 documents，以及分數最低的 kir 個 documents。在 Rocchio 中，Query、相關文件和不相關文件，有各自的權重，我以(q, r, ir)表示。我將 feedback 次數設為 2，並且設計了一個收斂的機制，早一步剔除明顯不相關的文件。

完整的 VSM + Rocchio 機制如下：

1. 拿 query 對 46972 個 documents 做 VSM。
2. 取 kr(kir)個相關（不相關）文件做 Rocchio，得到第二版 query。
3. 拿第二版 query 對 2500 個 documents 做 VSM。
4. 取 kr(kir)個相關（不相關）文件做 Rocchio，得到第三版 query。

5. 拿第二版 query 對 2500 個 documents 做 VSM。
6. 取前 100 名 documents 作為預測結果。

從實驗結果可以看出，加入了 Rocchio 後分數沒有比單純的 VSM 高。更明顯的是，當加入了不相關文件後，結果非常糟糕，可以呼應課堂所說的，不相關文件其實並沒有那麼重要。而將 query 的比重調低時，結果會稍微變差，顯示原始 query 對最終結果的重要性。

(kr, kir)	(q, r, ir)	Public score
(10, 0)	(0.8, 0.2, 0)	**0.76416**
(10, 0)	(0.6, 0.4, 0)	0.74048
(8, 2)	(0.8, 0.1, 0.1)	0.00698
(8, 2)	(0.6, 0.3, 0.1)	0.09900
(8, 2)	(0.6, 0.2, 0.2)	0.02459
(8, 2)	(0.6, 0.1, 0.3)	0.00422

三、心得

這次作業讓我知道，不要對題目一知半解就開始亂刻程式碼。一定要先想清楚，並且設計出能滿足需求的資料結構，否則所有程式都會立基於最一開始隨便寫寫的程式，就會跑的很慢。我之所以會設計一個收斂的機制就是因為經過 Rocchio 的 query 向量會變很長，跑得非常慢，為了加快就讓 VSM 要處理的 document 數量依次遞減。另外，我感受到 Okapi 的強大，原本 0.48 的成績，只是實作 Okapi 之後，瞬間衝到排行榜前幾名，果然是前人智慧的結晶。而 Rocchio 在這次作業給的 data 並沒有非常好的效果。