Web Retrieval and Mining Spring 2020 Programming HW1

B06507002 材料三 林柏勳

1. Describe your VSM (e.g., parameters….)

我的VSM是使用bi-gram建立而成，term的選取是從query內的title和concept去抓取，抓取方式敘述如下:將concept中以逗點分隔的詞彙分別處理成term，如果這個詞彙大於三個字，那麼就兩兩拆分，如:流浪狗這個詞彙會被分為流浪和浪狗兩個term。在title中，則是先用jeiba進行分割後，再用自製的stopword列表去除stopwords，之後如同concept內詞彙，直接納入term，或是拆分後再納入term。接著依靠TF-IDF的方法、以及先前建立好的term的字典，分別建立query和documents的向量，documents資料選取是來自inverted-file進行抓取。

1. Describe your Rocchio Relevance Feedback (e.g., how do you define relevant documents, parameters…)

Relevance feedback 可以用下面的公式來實現

是原本query的向量，是和query相關文件的集合，則為和query不相關的文件數目。在本次作業中，我們沒辦法得到真正相關文件和不相關文件的集合，所以只好使用sudo-relevance feedback，取前k筆相關的文件，作為真正相關文件的集合，判斷剩下的皆屬於非相關文件。在這個模型中，可調的參數為我在實作上最後取k=20、，不過在這次的實驗中，我發現做relevance feedback會使結果變差。

1. Results of Experiments
2. MAP value under different parameters of VSM

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| term | query feature | k1 | k3 | b | 做SVD | MAP |
| 取inverted-file裡面所有的unigram | 4個皆取 | 1.2 | 1000 | 0.7 | 選擇eigenvalue最大的前1000維 | 0.65706 |
| 取inverted-file裡面所有的unigram，去除stopwords | 4個皆取 | 1.2 | 1000 | 0.7 | 選擇eigenvalue最大的前1000維 | 0.67002 |
| 取concept中的bigram | concept | 1.2 | 100 | 0.65 | 選取9成最大的eigenvalue | 0.60240 |
| 取concept中的bigram | concept | 1.2 | 100 | 0.65 | 無 | 0.75334 |
| 取concept、title中的bigram | concept、title，去除stopwords | 1.2 | 100 | 0.65 | 無 | 0.76641 |
| 取concept、title中的bigram | concept、title、question | 1.2 | 100 | 0.65 | 無 | 0.74288 |

1. Feedback vs. no Feedback

在本次作業中，我試著實作了sudo-relevance feedback，但是，無論怎麼調整參數，MAP score在validation的時候都會降低，因此，最後選取output file的時候，並不打算選擇做過relevance feedback的結果。下表中，r為做relevance feedback的次數，參數為k=20、α=0.95、β=0.05、γ=0。

|  |  |
| --- | --- |
| r | MAP (evaluation) |
| 0 | 0.754 |
| 1 | 0.739 |
| 2 | 0.72 |
| 3 | 0.694 |
| 4 | 0.683 |
| 5 | 0.668 |

1. Other experiments you tried

在本次實驗中，我試著調整了很多VSM的架構與參數，例如調整k1、b、k3的值、relevance feedback中 的值，以及SVD中選擇最大eigenvalue的數目或比例。不過，大部分的調整都沒有顯著的進步。比較有效的調整在於:

1. 製作dictionary時，選取term的方法:

一開始直接拿inverted-file內所有unigram來做為term，不但佔據內存很大的空間，做起來的效果有沒有很好，而且在算SVD的時候還要跑很久。

後來想到可以從query開始著手，因為只要找跟query很相似的字詞的document，就可以達到不錯的檢索成果，而且query的長度、詞彙量也沒有inverted-file來得多(大概差了1000倍)。而想要使用bi-gram的原因在於:做unigram很容易遇到noise，比方說: 「流浪狗」如果只選取「流」、「浪」、「狗」分開三個字，那麼「流」、「浪」可能會檢索到跟水相關的文件，如: 「流水」、「衝浪」等，但是改為檢索「流浪」、「浪狗」將可避免這樣的問題。

1. 選取query中feature

一開始直接拿全部的feature做query term vector，效果不佳，後來發現在narrative跟question中擁有太多不相關的文字，這些字會影響搜尋的結果，例如: 「相關文件內容」、「包括」等等，即便刪除這些文字結果也沒有變好，所以最後索性挑選最精練的concept，結果就被顯著的提升了，後來再加入: title，可以在進一步提升MAP值。

1. Discussion: what you learn in the homework.

我覺得我在這份作業中學到的是要怎麼處理大型的資料，一開始想暴力的讀取全部inverted-file的內容，結果導致內存不足的情況，後來嘗試降維、選擇跟query內容相關的term作為vector的候選，才能夠解決這樣的問題，這是一般的程式課中較少能夠學到的部分。而我也了解到，IR跟ML具有很大的不同點， ML可以靠著調參數顯著的提升performance，但是IR不行，不論怎麼調，結果都差不多，所以該下功夫的地方會在於data preprocessing的部分，而非最後VSM裡面參數的調整。

1. 使用套件

numpy, jeiba, pandas, sys, xml, pickle, argparse