Information Retrieval and Applications

HW05-Pseudo Relevance Feedback Model

M10915201陳牧凡

這次的作業需要實作的演算法PRF最大的特色在於考慮query的準確性，因為在做資料檢索中，information need 和query未必是一致的，因此我們需要將query進行修正，讓其能夠符合information need。我採取的修正方法是Rocchio演算法。

$$\vec{q}' = \alpha \cdot \vec{q} + \beta \cdot \frac{1}{|R_q|} \cdot \left(\sum_{d_j \in R_q} \overrightarrow{d_j} \right) - \gamma \cdot \frac{1}{|\bar{R}_q|} \cdot \left(\sum_{d_{j'} \in \bar{R}_q} \overrightarrow{d_{j'}} \right)$$

首先需要取得該query的ranking，已取得他的relevant&nonrlevant doc，那麼取得的方式事先做一次IR，plsa、bm25或vsm都可以。檢索完後取各個query的ranking後再透過調參數a,b,r取得較佳的q'解。這邊特別要提，由於要分別取得topK的relevant以及nonrelevant，而使用bm25及vsm做出來的IR結果相似度低的doc會出現一種現象就是沒有出現任何query的字，也就是說他們的分數是0分，而這樣的doc出現的數量很可能是數千甚至是超過一半非常的多，這會導致nonrelevant那一項得出的結果很不準確。因此，第一次的IR我是採用plsa的結果，以確保排序的出的分數不會出現0分(以機率的形式表示)，進而得到比較精確結果。

PLSA's parameters:topic=32,em_iteration=100,a = 0.7,b = 0.3

單純只使用PLSA算出來的排序上傳kaggle的話MAP分數會得到約**0.48**。

取的了第一次IR算出的ranking後，我將這個ranking序列前K個當作relevant，後K個當作nonrelevant，再套用上面的Rocchio演算法，最後可以得出修正過後的query。接下來就將新的query和document計算相似度(即第二次IR)。我使用的方法是之前做過的BM25，沿用第二次作業，我組合了BM25L、BM25以及BM1的分數，並透過各自的權重來調整，為方便稱呼我就叫他BM26(25+1)。

$$bm26_{similarity} = bm25L_{similarity} * a1 + bm25_{similarity} * a2 + bm1_{similarity} * (1 - a1 - a2)$$

透過不斷的調整各種參數(k1,k3,b,delta,...etc.)，可以在kaggle上得到略高於baseline分數的0.52，大致上都落在**0.52~0.53**這個區間。不過不管怎麼調這些參數，改進的幅度都相當有限，因此我想要作一些根本上的改變。

那我就想到在表示doc時，依照作業二原始的作法我是只取query出現過的字來建字典，那在使用rocchio演算法的情況下，他會將query和doc相加在一起，那如果只用原來的字典來做這件事的話他可能會無法清楚表達這個新query的意思，因此在這邊我會重建字典，來讓qeury及doc能以更多字來表示。新的字典我拿所有query的所有relevant doc來建，讓新的query可以代表更多的資訊。這也確實讓準確度得到了改善，kaggle上的MAP分數可以超過**0.53**，而且每次調整參數都能得到很大的差異，調了幾次參數MAP分數就來到**0.57~0.58**。

以下是我嘗試各種組合參數中最好的一組:

#rocchio

fix=5,n_fix=5,alpha=1,beta=0.5,gamma=0.3

#bm26

a1 = 0.6,a2 = 0.3,k1 = 1,k3 = 1000,b = 1,delta = 0.75

這樣得到的kaggle public分數是0.582。其中fix,n_fix代表Rq,Rnq的doc數量。這兩參數我有嘗試多組，其中我發現fix,n_fix設置不一樣會讓分數下降很多，但設置一樣的話就表現還不錯。另外數量的部分我有設過3,5,10，效果上差不多，不過平均上來說5的效果略好一點點，0.57跟0.58的差別。

那除了Rocchio algorithm我還有嘗試使用Simple Mixture Model(SMM)，一樣是先經過第一次的IR排序，得到每個query的relevant doc，那我用的跟上面一樣是PLSA。接下來使用SMM來得出新的query表示向量Psmm(w)，並可以仿照PLSA時結合unigram model以及background model，以這樣的式子來表示P(w|q)。如下:

$$[\alpha \cdot P_{ULM}(w) + \beta \cdot P_{RM}(w) + (1 - \alpha - \beta) \cdot P_{BG}(w)]$$

其中beta項的P_{RM}可以替換成任意SMM或TriMM等等的向量表示法，也就是說relevant、simple mixture以及tri-mixture model在最後輸出新query的時候是共用這個公式的。得到新query後，在使用KL divergence計算query、doc間的距離，算出各query、doc間的距離後越接近零的代表相關度越高:

$$-\sum_{w \in V} P(w|q) \log P(w|d_j)$$

雖然有做出來，但是算出來的結果不管怎麼試，上傳kaggle的複數都很糟，也不曉得是不是有誤解數學式子的意思或是寫錯了，所以就先放棄這個部分的實作結果了。