Information Retrieval and Applications

HW02-Best Match Model 25

M10915201陳牧凡

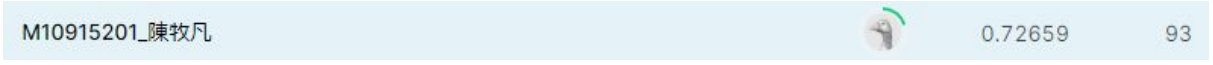
此次作業運用到的MODEL是BM25，其是融合BM11以及BM15並透過可調整參數進行調整。BM系列跟VSM最大的差別主要就在於BM會去考慮long document penalty，在考慮docuent TF時會除以long document normalization以抵銷長的文章所帶來的一些問題，像是一篇100字文章和一篇1000字文章，他們符合query的字數同樣都是20個字，理論上100字的那篇相關性應該要比另一篇來的高，但是依照一般TF的算法無法去區分兩者的優先順序。

BM25的公式簡單來說就是: $TF(doc) * TF(query) * IDF$ ， 而 $TF(doc)$ 和 $TF(query)$ 透過K1、K3、b三個參數進行調整， 且 $TF(doc)$ 會去考慮long document penalty。而在我自己嘗試的過程中，還另外做了BM25L以及BM1(其實就是只有IDF)，並調整各自權重最後再輸出結果:

BM score = a1*BM25L+a2*BM25+(1-a1-a2)*BM1

```
def calculateBM25L(k1, k3, b, delta, DLN, TFq, TFd, IDF):
    TFpron = TFd/(1-b+b*DLN)+delta
    F = (k1+1)*TFpron/(k1+TFpron)
    TF = (k3+1)*TFq/(k3+TFq)
    SIMbm25L = np.sum(F*TF*IDF*IDF)
    return SIMbm25L
def calculateBM25(k1, k3, b, delta, DLN, TFq, TFd, IDF):
    F = (k1+1)*TFd/(k1*(1-b+b*DLN)+TFd)
    TF = (k3+1)*TFq/(k3+TFq)
    SIMbm25 = np.sum(F*TF*IDF*IDF)
    return SIMbm25
def calculateBM1(TFq,IDF):
    return np.dot(IDF,TFq)
```

正確率是:72.66%



參數調整: k1 = 2.5, k3無所謂*, b = 0.8, BM25L's delta = 0.75
BM score權重: a1 = 0.7, a2 = 0.1

*k3是query的權重項，因為作業中的query都是短的query因此計算TF時有出現字的維度通常都是只有1，那當TFq=1時(K3+1)*TFq/(K3+TFq)中K3不管代入多少，最後結果都為1，因此不需要特別在意K3設定多少。