ASsignment 2

Liam Brock n9955038

QUT

CAB431 SEARCH ENGINE TECHNOLOGY

# Question 1

**Inputs:**   
Topic query ‘q’, Set of xml documents ‘U’

**Output:**   
A Training.txt file of lines following the format {Topic} {DocumentID} {Relevance}. The relevance score indicates if the given document is relevant given the topic query. A score of 1 indicates that the document is relevant and a score of 0 indicates that the document is not relevant.

**Method:**Step 1  
Process each xml document into a BowDoc object and store in a BowDocColl object.  
To convert the xml document into a BowDoc object:  
a) Extract text from <text> element and its children and preprocess into a list of terms  
b) Create a BowDoc object identified by the xml documents ‘itemid’ attribute  
c) Add all extracted terms to the BowDoc object

Step 2  
Calculate TFIDF score for each BowDoc in the BowDocColl for the given query  
a) Document adjusted TF = frequency/doc length  
b) IDF = log(num docs/1+document frequency)  
c) TFIDF = TF\*IDF

Step 3  
Sort TFIDF scores in descending order and save into relevant ‘Training.txt’ file

# Question 4

## Training Algorithm

The training algorithm used is the ‘BM25termWeighting’ algorithm from the lectures.  
The implementation is slightly different to reuse code. For example, ntk is calculated using the get\_df method of the BowDocColl object. T is also calculated in the same method except a new BowDocColl is first created with only relevant BowDocs.

**Inputs:**  
BowDocColl ‘coll’, Training file read into a dictionary ‘D’, experimental parameter ‘theta’

**Output:**  
A dictionary ‘features’ containing keyword : score pairs.

**Method:**  
Step 1:  
Calculate T:  
a) Create a new BowDocColl ‘dpos’ containing only relevant BowDocs using ‘D’  
b) T = dpos.get\_df()  
  
Step 2:  
Calculate ntk: ntk = coll.get\_df()

Step 3:  
Calculate R. R = the number of relevant documents in coll  
Calculate n. n = the number of documents in coll  
  
Step 4:  
For each term in T calculate the score where:  
rtk = Frequency of term in relevant document collection ‘dpos’ = T[term]  
ntk = Frequency of term in document collection ‘coll’ = ntk[term]  
a) score = ((rtk+0.5) / (R-rtk+0.5)) / ((ntk-rtk0.5) / (n-ntk-R+rtk+0.5))  
  
Step 5:  
Calculate the mean: Sum score of each value in T and divide by number of entries in T

Step 6:  
Calculate the features.  
a) If the score of the feature is greater than mean + theta it is included

## Testing Algorithm

**Inputs:**Collection of BowDoc ‘coll’, Features ‘features’

**Output:**  
A dictionary (sorted descending order) of docid : score pairs.  
  
**Method:**Step 1  
Create empty dictionary ‘ranks’  
  
Step 2For each docid, BowDoc in the collection do the following:  
a) Get term list of BowDoc ‘terms’ using method get\_terms()  
b) For each term in ‘features’  
 If term exists in terms then add features[term] to ranks[docid]

# Question 7

Using the t-test we can see that mean precision for the IR model is greater than the mean precision of the baseline model. It is observed that the t Stat is greater than both the t Critical one-tail value and t Critical two-tail value. This indicates that the IR Model has a greater average precision than that of the baseline model.

