Information Retrieval Lab 3

1. Objectives

- Write code to compute real TF and IDF values
- Recompute inverted index
- Match Query with Document

2. Compute TF and IDF

Write a function compute_tf() which computes the TF for a word in a given document, using the equation:

$cfreq_{ij} = K + (1 - K) (freq_{ij} / maxfreq_j)$

$freq_{ij}$	=	freq of term i in document j
$maxfreq_j$	=	max freq of any term in document j
K	=	*0.5 (let us assume)*

compute_tf() can take any parameters you wish.

Write a function compute_idf() which computes IDF for a word, using the equation:

$IDF_i = log_2 (N / n_i + 1)$

N	=	number of documents in collection
n_i	=	total occurrences of term *i* in collection

compute_idf() can take any parameters you wish.

Remember, the IDF of a word is always the same for a document collection. It does not vary from query to query.

3. Recompute Inverted Index

Recompute these values:

df - global variable for document frequency
idf - global for inverted document frequency
forward_index - global for forward index
inverted_index - global for inverted index

Note: These values are not normalised yet. Let us ignore that for now.

4. Match a query with a document

Write a function search_for_terms()

It should take one parameter which is a list of terms, e.g.

```
[ ['淘宝', 1 ], [ '诞生', 1 ] ]
```

(We assume the weight of each query term is 1)

It should return an ordered list of up to five document matches:

```
[ [ '双十一购物狂欢节_2017115146.htm', 1.2 ],
[ '淘宝小二_2017115146.htm', 0.9 ], ... ]
```

The highest match (1.2) comes first and the lowest match comes last in the list.

Note: Matches computed by your program may not be 'correct' and may not be ordered correctly by your system.

To compute the match of a query with a document, compute the dot product of the query vector with the document vector. e.g. suppose the TF*IDF weight for `'双十一购物狂欢节_2017115146.htm'` for the term `'淘宝'` is 0.8 and the TF*IDF weight for `'双十一购物狂欢节_2017115146.htm'` for the term `'诞生' is 0.4` . As shown above, the weight for `'淘宝'` in the query is 1 and the weight for `'诞生'` in the query is `0.4`. So the dot product is 1 * 0.8 + 1 * 0.8 = 1.2.

You can obtain the weights for a particular document by looking up the term in the inverted index.

5. Produce Sample Output

Take the first four queries from the gold standard and search for them using your search_for_terms() function. Write out the results as shown below.

6. Files and Functions

Call your program search_index.py
It can contain all the code from create_index.py plus some new functions to handle the searching.

The output you produce should show the document matches for the first four queries in the gold standard, like this:

```
Input:  [ ['淘宝', 1 ], [ '诞生', 1 ] ]
Output: [ [ '双十一购物狂欢节_2017115146.htm', 1.2 ],
[ '淘宝小二_2017115146.htm', 0.9 ], ... ]

Input:  [ ['淘宝', 1 ], ['交易', 1 ] ]
Output: [ ... ]

Input:  [ ['淘宝', 1 ], ['网上', 1 ], ['开店', 1 ] ]
Output: [ ...

Input: [ ['淘宝', 1 ], ['收入', 1 ] ]
Output: ...
```

Call the output file 'match.txt'

7. Upload your program to Moodle

Call your program file search_index.py (it should contain the create_index.py code with additions).
Call your output file match.txt
include tcf_tokenise_chinese_file.py

On Moodle, go to Week 4, i.e. 20 September - 26 September. Click Week 4 Lab 3. Upload your files search_index.py, tcf_tokenise_chinese_file.py and match.txt there.

Please check you have:

- Comment at the start (marks for this) with your name and number
- Code for the function search_for_terms() (use other functions / classes in addition as you wish)
- Output in match.txt as specified