# Assignment 3

In this assignment you will perform ranked retrieval, from index you create in assignment 2, and evaluate your results.
You will work in same group as in Assignment 2. You cannot change your group
Assigned on 23nd April, 2019
Due date: 1st May, 2019
Submission on Slate

## Overview

In this assignment, you will use the index you created in Assignment 1 to rank documents and create a search engine. You will implement several different scoring functions and compare their results against a baseline ranking produced by expert analysts.
Running Queries
For this assignment, you will need the following two files:

- **topics.xml** : contains the queries you will be testing. You should run the queries using the text stored in the <query> elements. The <description> elements are only there to clarify the information need which the query is trying to express.
- **corpus.qrel:** contains the relevance grades from expert assessors. While these grades are not necessarily entirely correct (and defining correctness unambiguously is quite difficult), they are fairly reliable and we will treat them as being correct here.

The format is:

<topic> 0 <docid> <grade>

Where

- <topic> is the ID of the query for which the document was assessed.
- 0 is part of the format and can be ignored.
- <docid> is the name of one of the documents which you have indexed.
- <grade> is a value in the set {-2, 0, 1, 2, 3, 4}, where a higher value means that the document is more relevant to the query. The value -2 indicates a spam document, and 0 indicates a non-spam document which is completely non-relevant. Most queries do not have any document with a grade of 4, and many queries do not have any document with a grade of 3. This is a consequence of the specific meaning assigned to these grades here and the manner in which the documents were collected.

This QREL does not have assessments for every (query, document) pair. If an assessment is missing, we assume the correct grade for the pair is 0 (non-relevant).

You will write a program which takes the name of a scoring function as a command line argument and which prints a ranked list of documents for all queries found in topics.xml using that scoring function. For example:

```
$ ./query.py --score TF-IDF
202 0 clueweb12-0000tw-13-04988 1 0.73 run1
202 0 clueweb12-0000tw-13-04901 2 0.33 run1
202 0 clueweb12-0000tw-13-04932 3 0.32 run1
...
214 0 clueweb12-0000tw-13-05088 1 0.73 run1
214 0 clueweb12-0000tw-13-05001 2 0.33 run1
214 0 clueweb12-0000tw-13-05032 3 0.32 run1
...
250 0 clueweb12-0000tw-13-05032 500 0.002 run1
```

The output should have one row for each document which your program ranks for each query it runs. These lines should have the format:

**<topic> 0 <docid> <rank> <score> <run>**

where

•<topic> is the ID of the query for which the document was ranked.
•0 is part of the format and can be ignored.
•<docid> is the document identifier.
•<rank> is the order in which to present the document to the user. The document with the highest score will be assigned a rank of 1, the second highest a rank of 2, and so on.
•<score> is the actual score the document obtained for that query.
•<run> is the name of the run. You can use any value here. It is meant to allow research teams to submit multiple runs for evaluation in competitions such as TREC.

Both of these files are given at following path
\\cactus\xeon\Spring 2019\Noshaba Nasir\Information Retrieval\Assignment3

## Query Processing

Before running any scoring function, you should process the text of the query in exactly the same way that you processed the text of a document.

## Scoring Functions

The --score parameter should take one of the following values, indicating how to assign a score to a document for a query.

1. Okapi TF

   Formula

$$\text{Score} = \sum_{t \in q} \left[ c_t \times \frac{(k_1+1)\text{tf}_{td}}{k_1((1-b)+b(L_d/L_{\text{ave}}))+\text{tf}_{td}} \right]$$

$$c_t = \log \frac{N - \text{df}_t + \frac{1}{2}}{\text{df}_t + \frac{1}{2}}.$$

   Where k1=1 and b=1.5 (you can try other values of these constants)

2. Vector Space

Formula

$$score(d) = \frac{\vec{d} \cdot \vec{q}}{\|\vec{d}\| \cdot \|\vec{q}\|}$$

Where $\|\vec{x}\| = \sqrt{\sum_i x_i^2}$ is the norm of vector $\vec{x}$.

The feature vector weighting scheme should be TF-IDF

## Evaluation

Use the NDCG to evaluate the results. Write a script named NDCG.py that will take the output if query.py and corpus.qrel file and p. This should return the NDCG score at position p for each query, and average NDCG score (i.e. average for all queries).

```
$ ./query.py --score vector-space > run.txt
$ ./NDCG.py corpus.qrel run.txt 1
```

Where run.txt is the file where output of query.py should be written.

Find the NDCG and Average NDCG at different values of p, for example p=1,3,5,10..., and report your results in form of table of graph.

## Submission:

Submit query.py, NDCG.py, run.txt file for each scoring technique. Report of results.