# IR Assignment-2

**-MT18117 SUBHANI SHAIK**

**Dataset:**

- Stories : it has documents which have text data with headings in different formats.
- So I loaded index.html files separately and extracted the required data into two files names_1.txt and names_2.txt(which has file names of directories 'stories' and 'SRE' respectively).
- Total 467 files are there and loaded for pre-processing.

**Pre-Processing the Dataset:**

To pre-process the data, I used library NLTK and done following steps:

- Converted to lowercase.
- Contractions are fixed.
- Removed unnecessary characters and symbols and tokenized by using RegexpTokenizer from NLTK.
- Done Lemmatization.
- Done Stemming using PorterStemmer from NLTK.

This pre-processing is done for training and to input(for testing queries) also.

**1. Tf_Idf based document retrieval:**

***Tf-Idf = Term Frequency x Inverted Document Frequency***

The query terms may present in any where in the document like title zone or body zone.

Here to reflect the importance of word in zones, I have given weights to those words along with Tf-Idf value.

$$\textbf{Tf-Idf=weight*(tf/N)*log}(N_d/df)$$

Where

tf= frequency of a term in the document

N= number of words in the document

$N_d$ = number of documents in the collection

df = frequency of the document

- Initially I constructed dictionary for titles and for body separately and created inverted index for document data.
- Given a query, the documents containing all the terms is found by the intersection of documents that contain individual tokens of the query. Intersection returns the common documents in which all the terms present in the query occur.
- Now, the tf-idf value of each term in the query is calculated w.r.t the intersection of documents.
- If the term occurs in the heading as well as the document, higher weight is assigned to the tf-idf. Similarly, if the term occurs in the heading, an additional weight, but not greater than the previous case is assigned. Otherwise, a lower weight is assigned.
- The tf-idf is now calculated taking into the above considerations.
- The documents are now ranked in sorted order of tf-idf value i.e., the document importance w.r.t the query.

Results:

Input 1: “trip going”

Output:

```
['stories/quarter.c15', 0.0035585896705922822]
['stories/bgb.txt', 0.002587547133359512]
['stories/crabhern.txt', 0.0020393773750262034]
['stories/arctic.txt', 0.0013388744834127143]
['stories/blind.txt', 0.0012635230228393952]
```

Input 2: “This is information”

Output: filename, tf-idf value

```
['stories/sanpedr2.txt', 0.0024820681502728823]
['stories/mindprob.txt', 0.002372439893947009]
['stories/tin', 0.002367507936615835]
['stories/jaynejob.asc', 0.0017728833098760938]
['stories/campfire.txt', 0.0017305107547000544]
['stories/bran', 0.0016537178322982892]
['stories/SRE/srex.txt', 0.0016371626296634678]
```

**2. Tf-Idf based Vector space document retrieval on Cosine similarity:**

- Given a query, the documents containing all the terms is found by the intersection of documents that contain individual tokens of the query. Intersection returns the common documents in which all the terms present in the query occur.
- Now, the term frequency of each term in the query is calculated w.r.t the intersection of documents.
- For each document in the intersection, the cosine similarity is calculated between the document vector and the query vector which is given by

$$Cos(\theta)=(d_j.q) / ||d_j||*||q||$$

Where

$d_j$ =document j vector

q=query vector

$|| d_j ||$ =magnitude of document vector

||q||=magnitude of query vector

- The documents are now ranked in order of cosine similarity w.r.t the query.

Results:

Input 1: “trip going”

Output 1: filename, cosine similarity value

```
['stories/quarter.c15', 0.07570789571609836]
['stories/bgb.txt', 0.07281336085277718]
['stories/chik', 0.057854396305657]
['stories/kzap.txt', 0.04512025564538061]
['stories/dan', 0.0415962757434186]
```

Input 2: “This is information”

Output :

```
['stories/dakota.txt', 0.18468984989631773]
['stories/keepmodu.txt', 0.18216997471626903]
['stories/tin', 0.17004038938928964]
['stories/enya_trn.txt', 0.16040993324170455]
['stories/blabnove.txt', 0.1453337817731396]
['stories/mindprob.txt', 0.13605159725331115]
['stories/jaynejob.asc', 0.13046637931827693]
```