**Information Retrieval using vector space model**

Algorithm for text Information Retrieval engine:

1. Read a file
2. Extract All "DOC" tags and its data
3. Give Extracted data to text processing module

**Task of Text Processing Module**:

* 1. Remove all punctuations
  2. Tokenization
  3. Remove numbers and alphanumeric strings
  4. Remove stopwords
  5. stem words using PorterStemmer

**Task for indexer**

1. Forward indexing
2. invert indexing
3. By using above tokens, forward indexing and invert indexing find tf\*idf score
4. After getting TF-IDF score using that create vector from that
5. Now take Query convert it into tokens using text processing module, find create vector form of query (using tf-idf) and find cosine similarity score between query and documents
6. Store results of query in output file.

**title\_similarity\_output.txt** : contains similarity score for documents and title from query

**title\_desc\_similarity\_output.txt** : contains similarity score for documents and title + description from query

**title\_narrative\_similarity\_output.txt** : contains similarity score for documents and + narrative from query

**Function Understanding:**

* 1. Extract\_query\_df() :
     1. Input parameter : none
     2. Task : This function reads topics.txt, remove tags and store data in data frame
     3. Output : return dataframe which has column

(query\_number,title,description,narrative)

* 1. tf\_idf\_score(N,processed\_data,data\_freq):
     1. Input parameter :
        1. N : total number of documents
        2. Processed\_data :document and token dictionary
        3. Data\_freq : data frequency dictionary which contains token and how much document it appeared
     2. Task : provide tf\*idf score for each document
     3. Output : tf\*idf score for each document
  2. cosine\_similarity(N,document\_list,query,D,total\_vocab,data\_freq,query\_id):
     1. Input Parameter:
        1. N : total number of documents
        2. Document\_list : list of document id
        3. Query : query string
        4. D : vector of all document
        5. Total\_vocab : vocabulary (list of all tokens)
        6. Data\_freq : data frequency dictionary which contains token and how much document it appeared
        7. Query\_id : query number
     2. Task : Provide cosine similarity between query and documents
     3. Output : returns dataframe of cosine similarity with document and query
  3. gen\_vector(N,tokens,total\_vocab,data\_freq):
     1. Input Parameter:
        1. N : total number of documents
        2. Tokens : clean token of query
        3. Totak\_vocab : vocabulary (list of all tokens)
        4. Data\_freq : data frequency dictionary which contains token and how much document it appeared
     2. Task : Provide vector form of query
     3. Output : returns query in vector form
  4. cosin\_sim(a,b):
     1. Simple formula of cosine similarity
  5. rank\_document(result\_object):
     1. By using all above function rank the document and store data in output file