Setting up the environment

- Download the dataset from here
- For setting up the elastic instance I suggest to use a docker image.
- Make sure you have docker installed.
- Run the below command. This will launch an elastic instance
  `docker run -p 8200:9200 -p 8600:9600 -e "discovery.type=single-node" amazon/opendistro-for-elasticsearch:1.8.0`
- Download the sentence encoder from here
- I have used the opendistro's elastic search as they provide inbuilt Approximate Nearest Neighbors implementation. It uses nmslib's implementation for HNSW
- HNSW tops the benchmarks

Importing libraries

In [1]:
```
import numpy as np
import pandas as pd
from tqdm import tqdm
import json
import time
import sys
from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
import csv
import tensorflow as tf
import tensorflow_hub as hub
import warnings
from flask import jsonify
from flask import Flask
warnings.filterwarnings('ignore')
```

Connecting to the elastic instance

In [2]:
```
def connectES():
    '''
    This function creates a connection to the elastic search instance
    we provide the appropriate host name, port number, and authentication
    details.

        '''

    es = Elasticsearch([{'host': 'localhost', 'port': 8200,'use_ssl':True,'verify_certs':False}], http_auth=('admi
n', 'admin'))
    if es.ping():
        print('Connected to ES!')
    else:
        print('Could not connect!')
    return es
```

Loading the universal sentence encoder model

In [3]:
```
embed = hub.load("./use4")
```

Defining the index schema for elastic search

In [4]:
```
def createSchema(es):
    '''
        Creates a new index in the elastic search.
        Defines appropriate schema for the index.
        We define
            title       => text
            title_vector  => knn_vector

    '''

    #Refer: https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html
    # Mapping: Structure of the index
    # Property/Field: name and type

    b = {
      "settings": {
        "index": {
          "knn": True,
          "knn.space_type": "cosinesimil"
        }
      },
      "mappings": {
        "properties": {
            "title":{
                "type":"text"
            },
          "title_vector": {
            "type": "knn_vector", # Helps to find approximate k nearest neighbours
            "dimension": 512
          }

        }
      }
    }

    ret = es.indices.create(index='questions-index', ignore=400, body=b)
    print(json.dumps(ret,indent=4))
```

Data ingestion

In [5]:
```
def dataIngestion(es,NUM_QUESTIONS_INDEXED):
    '''
    This function helps us in ingesting the data into the
    elastic search index
    Here we index the title vector and the title.
    '''

    start_time = time.time()

    # Col-Names: Id,OwnerUserId,CreationDate,ClosedDate,Score,Title,Body
    cnt=0

    with open('Questions.csv', encoding="latin1") as csvfile:
        readCSV = csv.reader(csvfile, delimiter=',')
        next(readCSV, None)  # skip the headers
        for row in (readCSV):
            doc_id = row[0];
            title = row[5];
            body = row[6]
            vec = tf.make_ndarray(tf.make_tensor_proto(embed([title])).tolist()[0]

            b = {
                  "title":title,
                  "title_vector":vec,
                }


            res = es.index(index="questions-index", id=doc_id, body=b)

            cnt += 1
            if cnt%1000==0:
                print(cnt)


            if cnt == NUM_QUESTIONS_INDEXED:
                break;

        print("Completed indexing...")
    print("--- %s seconds ---" % (time.time() - start_time))
    print('**********************************************************************************')
```

Search engine implementation

In [6]:
```
#Search by Keywords
def keywordSearch(es, q):
    '''
    Implements the traditional keyword search using inverted index.
    Elastic search uses a TF-IDF based metric to rank the results.

    '''

    b={

            'query':{
                'match':{
                    "title":q
                }
            }

        }
    res= es.search(index='questions-index',body=b)
    return res
```

In [7]:
```
# Search by Vec Similarity
def sentenceSimilarityByNN(es, sent):
    '''
    Implements an approximate nearest neighbours method
    to find the k nearest vectors.
    Elastic search relies nmslib for the implementation of HNSW

    https://github.com/nmslib/hnswlib
    '''

    query_vector = tf.make_ndarray(tf.make_tensor_proto(embed([sent])).tolist()[0]
    kb={
        "size": 10,
        "query": {
          "knn": {
            "title_vector": {
              "vector": query_vector,
              "k": 10
            }
          }
        }
      }

    res= es.search(index='questions-index',body=kb,request_timeout=100)

    return res;
```

In [8]:
```
def searchEngine():
    '''
        Search engine method which
        shows results based on semantic similarity
        as well as keyword based similarity
        and shows the results in decreasing order of relevance
    '''
    es=connectES()
    print("Please give the input query")
    query=input()
    print('----------------------------SEMANTIC RESULTS--------------------------')
    res= sentenceSimilarityByNN(es,query)
    print("Semantic Similarity Search:\n")
    for hit in res['hits']['hits']:
        print(str(hit['_score']) + "\t" + hit['_source']['title'] )

    print('')

    res= keywordSearch(es,query)
    print('\n----------------------------KEYWORDS RESULTS--------------------------')
    for hit in res['hits']['hits']:
        print(str(hit['_score']) + "\t" + hit['_source']['title'] )
```

In [9]:
```
searchEngine()
```

```
Connected to ES!
Please give the input query
delete file in linux
----------------------------SEMANTIC RESULTS--------------------------
Semantic Similarity Search:

0.8078813       Removing a file in a Restricted Folder in Linux
0.78672296      remove certain tag in files under linux?
0.780427        Mailog file in linux
0.7667524       Bash: Delete until a specific file
0.75369155      remove directory in c++
0.7533658       Delete a line from a file in java
0.7423169       Delete unused files
0.7415796       Deleting files in higher directory
0.74069196      Remove a symlink to a directory
0.7325041       How to make files in Linux folder with default group write permission

('\n----------------------------KEYWORDS RESULTS--------------------------')
12.563389       Mailog file in linux
11.8293328      File paths in Java (Linux)
11.598585       Linux File Logs
11.566038       Delete an sdf file in use?
11.176312       md5sum of file in Linux C
10.960959       Delete file without playing sound in Applescript?
10.746357       My delete function does not delete the targeted file
10.6557045      Tortoise Delete File System Repository
10.416043       Delete a line from a file in java
10.416043       Delete a character from a file in C
```

Deploying using flask

In [10]:
```
def start_server():
    '''
        This function exposes our search engine using flask.
        The server is hosted on localhost:5005
        We perform GET request
        http://localhost:5005/search/recursion+vs+iteration
        use '+' as a seperator
    '''
    app = Flask(__name__)
    es = connectES();
    # embed = hub.load("./data/USE4/")

    @app.route('/search/<query>')
    def search(query):
        q = query.replace("+", " ")
        res_kw = keywordSearch(es, q)
        res_semantic = sentenceSimilarityByNN( es, q)

        result=[]
        result.append('----------------------SEMANTIC RESULTS-----------------------')
        start_time = time.time()

        for i in res_semantic['hits']['hits']:
            result.append(i['_source']['title'])

        result.append('----------------------KEYWORDS RESULTS-----------------------')

        for i in res_kw['hits']['hits']:
            result.append(i['_source']['title'])


        timeTaken = time.time() - start_time
        return {"result":result,"time_taken":timeTaken}
        return {"message": "Hello World"}

    app.run(host='0.0.0.0',port=5005)
```

Complete pipeline

In [11]:
```
def final():
    '''
        This runs the entire pipeline for the project.
        Right from data ingestion to exposing the API

    '''
    es=connectES()
    createSchema(es)
    dataIngestion(es,100) # number of documents to be ingested
    print("Status 400 denotes index already exists")
    print("Enter a search query")
    start_server()
```

In [12]:
```
final()
```

```
Connected to ES!
{
    "error": {
        "root_cause": [
            {
                "type": "resource_already_exists_exception",
                "reason": "index [questions-index/AZJN_HWBRZ2Zjs69ALoUoQ] already exists",
                "index_uuid": "AZJN_HWBRZ2Zjs69ALoUoQ",
                "index": "questions-index"
            }
        ],
        "type": "resource_already_exists_exception",
        "reason": "index [questions-index/AZJN_HWBRZ2Zjs69ALoUoQ] already exists",
        "index_uuid": "AZJN_HWBRZ2Zjs69ALoUoQ",
        "index": "questions-index"
    },
    "status": 400
}
Completed indexing...
--- 2.7990684509277344 seconds ---
**********************************************************************************
Status 400 denotes index already exists
Enter a search query
Connected to ES!
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5005/ (Press CTRL+C to quit)
127.0.0.1 - - [07/Jul/2020 22:33:16] "GET /search/Deleted+files+in+Linux HTTP/1.1" 200 -
127.0.0.1 - - [07/Jul/2020 22:33:28] "GET /search/aaaaaaa HTTP/1.1" 200 -
```