Final Python File

In [11]:

```
!ls
```

```
model.h5  Modified_SQL_Dataset.csv  sample_data  tflite_quant_model.tflite
```

Import Libraries

In [24]:

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import joblib
from sklearn.metrics import  roc_auc_score
import re
import nltk
nltk.download('punkt')
nltk.download('wordnet')
import string
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve,auc
from nltk.stem.porter import PorterStemmer
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from gensim.models import word2vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
from nltk.corpus import stopwords
import numpy as np
import tensorflow as tf
import tensorflow
import nltk
nltk.download('stopwords')
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
import numpy as np
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Import Dataset

In [13]:

```
data = pd.read_csv("Modified_SQL_Dataset.csv")
```

In [14]:

```
print(data.shape)
```

```
(30919, 2)
```

Import Train Test DataSet

In [15]:

```
data.drop_duplicates(inplace=True)
```

In [16]:

```
stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

In [17]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_query = []
lemmatizer = WordNetLemmatizer()
# tqdm is for printing the status bar
for sentance in tqdm(data['Query'].values):
    sentance = re.sub('[^A-Za-z0-9]+', ' ', sentance)
    sentance = re.sub(r',', ' ', sentance)
    #https://www.machinelearningplus.com/nlp/lemmatization-examples-python/
    tokenization = nltk.word_tokenize(sentance)
    sentance = ' '.join([lemmatizer.lemmatize(w) for w in tokenization])
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in stopwords)
    preprocessed_query.append(sentance.strip())
```

```
100%|██████████| 30907/30907 [00:05<00:00, 6085.21it/s]
```

In [18]:

```
max_length = max([len(s.split()) for s in preprocessed_query])
```

In [19]:

```
#vectorizer the text samples into 2d integer tensor
tokenizer_obj = Tokenizer()
tokenizer_obj.fit_on_texts(preprocessed_query)
sequences = tokenizer_obj.texts_to_sequences(preprocessed_query)

#pad_sequences

word_index = tokenizer_obj.word_index
print('found %s unique tokens.'%len(word_index))

query_pad = pad_sequences(sequences,maxlen=max_length)
label = data['Label'].values
print("shape of query_pad",query_pad.shape)
print("Shape of label",label.shape)
```

```
found 24489 unique tokens.
shape of query_pad (30907, 522)
Shape of label (30907,)
```

In [10]:

```
#split the data into train and test data

validation_split=0.2
indices = np.arange(query_pad.shape[0])
np.random.shuffle(indices)
query_pad = query_pad[indices]
label = label[indices]

num_validation_samples = int(validation_split*query_pad.shape[0])

X_train_pad = query_pad[:-num_validation_samples]
y_train = label[:-num_validation_samples]
X_test_pad = query_pad[-num_validation_samples:]
y_test = label[-num_validation_samples:]
```

Final Function 1

In [20]:

```
from tensorflow.keras.models import load_model
def final_fun_1(X):

  loaded_model = load_model('model.h5')
  y_pred = loaded_model.predict(X_test_pad)
  return y_pred
final_fun_1(X_test_pad)
```

Out[20]:

```
array([[3.8426219e-08],
       [1.0245228e-01],
       [2.1967155e-06],
       ...,
       [1.0245228e-01],
       [9.9999368e-01],
       [1.9717445e-06]], dtype=float32)
```

Final Function 2

In [21]:

```
def final_fun_2(X,Y):
    Y_pred = final_fun_1(X)
    score = roc_auc_score(Y, Y_pred)
    return score
final_fun_2(X_test_pad,y_test)
```

Out[21]:

```
0.9926657577602499
```

In [22]:

```
loaded_model = load_model('model.h5')
score  = loaded_model.evaluate(X_test_pad,y_test)
```

```
194/194 [==============================] - 60s 302ms/step - loss: 0.0656 - accuracy: 0.9790
```

In [23]:

```
print("Test loss {:.4f}, accuracy {:.2f}%".format(score[0], score[1] * 100))
```

```
Test loss 0.0656, accuracy 97.90%
```

In [26]:

```
import os
print("Float model in Mb:", os.path.getsize('tflite_model.tflite') / float(2**20))
print("Quantized model in Mb:", os.path.getsize('tflite_quant_model.tflite') / float(2**20))
print("Compression ratio:", os.path.getsize('tflite_model.tflite')/os.path.getsize('tflite_quant_model.tflite'))
```

```
Float model in Mb: 29.13034439086914
Quantized model in Mb: 7.293365478515625
Compression ratio: 3.9940881170263065
```

Post Training Quantization

In [25]:

```
#ref : https://stackoverflow.com/questions/65601060/tensorflowlite-error-interpreter-set-tensor/65675158#65675158
# Load TFLite model and allocate tensors.
interpreter = \
tf.lite.Interpreter(model_path="tflite_quant_model.tflite")
interpreter.allocate_tensors()
# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
# Test model on some input data.
input_shape = input_details[0]['shape']
acc=0
for i in range(len(X_test_pad)):
    input_data = X_test_pad[i].reshape(input_shape)
    input_data = input_data.astype(np.float32)
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    output_data = interpreter.get_tensor(output_details[0]['index'])
    if(np.argmax(output_data) == np.argmax(y_test[i])):
        acc+=1
acc = acc/len(X_test_pad)
print(acc*100)
```

```
100.0
```