```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
"""
# This Python 3 environment comes with many helpful analytics librar
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_cs

# Input data files are available in the read-only "../input/" direct
# For example, running this (by clicking run or pressing Shift+Enter

import os
for dirname, _, filenames in os.walk('path goes here..'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
"""
```

```
'\n# This Python 3 environment comes with many helpful analytics libraries installed\n#
For example, here\'s several helpful packages to load\n\nimport numpy as np # linear al
gebra\nimport pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)\n\n# Inpu
t data files are available in the read-only "../input/" directory\n# For example, runni
ng this (by clicking run or pressing Shift+Enter) will list all files under the input d
```

```
pip install stop_words
```

```
Collecting stop_words
  Downloading https://files.pythonhosted.org/packages/1c/cb/d58290804b7a4c5daa42abbbe2a
Building wheels for collected packages: stop-words
  Building wheel for stop-words (setup.py) ... done
  Created wheel for stop-words: filename=stop_words-2018.7.23-cp36-none-any.whl size=32
  Stored in directory: /root/.cache/pip/wheels/75/37/6a/2b295e03bd07290f0da95c3adb9a74b
Successfully built stop-words
Installing collected packages: stop-words
Successfully installed stop-words-2018.7.23
```

```
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import re
import nltk
```

```
#from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from sklearn import preprocessing
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, MaxPool1D, Dropout, Dense,
from keras.utils import to_categorical
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from stop_words import get_stop_words

```

Importing dataset , dropping unnamed columns and ceeating a test data

```
train_data_path = '/content/gdrive/My Drive/Colab Notebooks/Hate Spe
train_data = pd.read_csv(train_data_path + 'train.csv')

train_data = train_data.loc[:, ~train_data.columns.str.contains('^Un

print('original train data shape',train_data.shape)

#test_data = train_data.iloc[137571:,:]
#test_data = test_data.reset_index()
#train_data = train_data.iloc[:137570,:]

#print('test data shape',test_data.shape)
#print('new train data shape',train_data.shape)
#print(test_data.shape[0]+train_data.shape[0])
```

```
original train data shape (159571, 8)
```

```
train_data.head(5)
```

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	ider
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	
		D'aww! He						

Preprocessing the dataset

```
def deleteSmallWords(text):
    return ' '.join([word for word in text.split() if len(word) > 3]
def cleanText(text):
    # clean the text
    text = re.sub(r"Https?://[A-Za-z0-9./]+","url",text)
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]"," ",text)
    text = re.sub(r"what's","what is ",text)
    text = re.sub(r"\'s"," ",text)
    text = re.sub(r"\s+[a-zA-Z]\s+", ' ', text) # Single character r
    text = re.sub(r"\'ve"," have ",text)
    text = re.sub(r"\n't"," not ",text)
    #text = re.sub(r"\i'm","i am ",text)
    text = re.sub(r"\'re"," are ",text)
    text = re.sub(r"\'d"," would ",text)
    text = re.sub(r"\'ll"," will ",text)
    text = re.sub(r"\."," ",text)
    text = re.sub(r"!"," ",text)
    text = re.sub(r"\/"," ",text)
    text = re.sub(r"\^"," ^ ",text)
    text = re.sub(r"\+"," + ",text)
    text = re.sub(r"\-"," - ",text)
    text = re.sub(r"\="," = ",text)
    text = re.sub(r":"," : ",text)
    text = re.sub(r"'"," ",text)
    text = re.sub(r"(\d+)(k)",r"\g<1>000",text)
    text = re.sub(r" e g "," eg ",text)
    text = re.sub(r" b g "," bg ",text)
    text = re.sub(r" u s "," amarican ",text)
    text = re.sub(r"\0s","0",text)
    text = re.sub(r" 9 11 ","911",text)
    text = re.sub(r"e - mail","email",text)
    text = re.sub(r"j k","jk",text)
    text = re.sub(r"\s{2,}"," ",text)
```

```
        text = re.sub(r"@[A-Za_z0-9]+",'',text)
        text = re.sub(r"(\w)\1{2,}",r"\1\1",text)
        text = re.sub(r"\w(\w)\1{2}",'',text)
        return text
    def deleteNonAlphaWords(text):
        return ''.join([word for word in text.split() if word.isalpha()]
    def deleteStopWords(text):
        return ' '.join([word for word in text.lower().split() if not wo
```

```
train_data['comment_text'] = train_data['comment_text'].apply(lambda
train_data['comment_text'] = train_data['comment_text'].apply(lambda
train_data['comment_text'] = train_data['comment_text'].apply(lambda
train_data['comment_text'] = train_data['comment_text'].apply(lambda

#test_data['comment_text'] = test_data['comment_text'].apply(lambda
#test_data['comment_text'] = test_data['comment_text'].apply(lambda
#test_data['comment_text'] = test_data['comment_text'].apply(lambda
#test_data['comment_text'] = test_data['comment_text'].apply(lambda
```

Tokenize the data

```
num_texts = len(train_data.index)
print(num_texts)
token = Tokenizer(num_words=num_texts)
token.fit_on_texts(train_data['comment_text'])
text = token.texts_to_sequences(train_data['comment_text'])
text = pad_sequences(text)

tt = 'bitch hate nigga'
text_test = token.texts_to_sequences(tt)
text_test = pad_sequences(text_test)
print(type(text_test))
```

```
159571
<class 'numpy.ndarray'>
```

```
columns = train_data.columns
columns = list(columns[2:])
print(columns)
#y = train_data.loc[:,columns].values
```

```
y = train_data['toxic']
print(text.shape,y.shape)
print(text[1],y[0])

```

```
['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']
(159571, 1) (159571,)
[1015] 0
```

```
x_train, x_test, y_train, y_test = train_test_split(text, y, test_si
print(x_train.shape,x_test.shape)
print(y_train.shape,y_test.shape)
print(type(text))
```

```
(127656, 1) (31915, 1)
(127656,) (31915,)
<class 'numpy.ndarray'>
```

```
max_features = num_texts
embedding_dim = 32

model = Sequential()
model.add(Embedding(max_features, embedding_dim))
model.add(Dropout(0.2))
model.add(LSTM(32, return_sequences=True))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 32)          5106272
_________________________________________________________________
dropout (Dropout)            (None, None, 32)          0
_________________________________________________________________
lstm (LSTM)                  (None, None, 32)          8320
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 32)          0
_________________________________________________________________
dense (Dense)                (None, None, 1)           33
_________________________________________________________________
activation (Activation)      (None, None, 1)           0
=================================================================
Total params: 5,114,625
Trainable params: 5,114,625
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
# compile and train model
print(x_train.shape,y_train.shape)
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=
history = model.fit(x_train, y_train, validation_data=(x_test, y_tes
```

```
(127656, 1) (127656,)
Epoch 1/10
3990/3990 [==============================] - 227s 57ms/step - loss: 0.3282 - accuracy: 0
Epoch 2/10
3990/3990 [==============================] - 238s 60ms/step - loss: 0.0510 - accuracy: 0
Epoch 3/10
3990/3990 [==============================] - 227s 57ms/step - loss: 0.0033 - accuracy: 0
Epoch 4/10
3990/3990 [==============================] - 229s 57ms/step - loss: 0.0024 - accuracy: 0
Epoch 5/10
1377/3990 [=========>....................] - ETA: 2:28 - loss: 0.0016 - accuracy: 0.9996
```

```
#model.predict(text_test)
```