```
import numpy as np
import re
import pickle
import nltk
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
en_stop = set(nltk.corpus.stopwords.words('english'))
from keras import backend as K
from keras.preprocessing.text import Tokenizer
from keras.models import load_model
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
Using TensorFlow backend.
```

The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the `%tens`

```
# Preprocess text
def preprocess_text(document):
    #now = datetime.datetime.now()

    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(document))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixed 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()

    tokens = document.split()
```

```
    #### Remove stopwords
    words = [w for w in tokens if w not in stopwords.words('english')]
    words = [word for word in words if word not in en_stop]

    #### Lemmatize tokens obtained after removing stopwords
    wnl = WordNetLemmatizer()
    tagged = nltk.pos_tag(words)
    lem_list = []
    for word, tag in tagged:
        wntag = tag[0].lower()
        wntag = wntag if wntag in ['a', 'r', 'n', 'v'] else None
        if not wntag:
            lemma = word
        else:
            lemma = wnl.lemmatize(word, wntag)
        lem_list.append(lemma)

    #preprocessed_text = ' '.join(lem_list)

    return lem_list


# Define functions to be able to calculate additional metrics like precision,recall,

def recall_m(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
        recall = true_positives / (possible_positives + K.epsilon())
        return recall

def precision_m(y_true, y_pred):
        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
        precision = true_positives / (predicted_positives + K.epsilon())
        return precision

def f1_m(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)
    return 2*((precision*recall)/(precision+recall+K.epsilon()))
def hamming_loss(y_true, y_pred):
  return K.mean(y_true*(1-y_pred)+(1-y_true)*y_pred)


# load model from single file
dependencies = {'f1_m': f1_m, 'recall_m':recall_m, 'precision_m':precision_m, 'hammin
toxic_lstm_single_10 = load_model('toxic_lstm_single_10.h5', custom_objects= dependen
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

/usr/local/lib/python3.6/dist-packages/keras/engine/saving.py:310: UserWarning: N
  warnings.warn('No training configuration found in save file: '
```

```
# load model from single file
dependencies = {'f1_m': f1_m, 'recall_m':recall_m, 'precision_m':precision_m, 'hammin
movie_lstm_single_10 = load_model('movie_lstm_single_10.h5', custom_objects= dependen
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/optimizers.p

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/py
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tens
```

```
# loading
with open('tokenizer_toxic_final.pickle', 'rb') as handle:
    tokenizer_toxic = pickle.load(handle)
```

```
# loading
with open('tokenizer_movie.pickle', 'rb') as handle:
    tokenizer_movie = pickle.load(handle)
```

```
def predict_comment(new_data,tokenizer,model_file):
  CATEGORIES = ["toxic","severe_toxic", "obscene", "threat", "insult","identity_hate"
  new_data = preprocess_text(new_data)
```

```
    new_data = preprocess_text(new_data)
    X_tokenized = tokenizer.texts_to_sequences(new_data)
    to_pad = []
    for x in X_tokenized:
        if len(x) >0:
            to_pad.append(x[0])

    while len(to_pad) < 200:
        to_pad.append(0)

    test = [x for x in to_pad]
    test = [test]
    test = np.array(test)

    prediction = model_file.predict(test)
    pred_name = CATEGORIES[np.argmax(prediction)]
    return pred_name, prediction


def predict_genre(new_data,tokenizer,model_file):
    CATEGORIES = ["Drama", "World cinema", "Action", "Black-and-white", "Romance Film",
    new_data = preprocess_text(new_data)
    X_tokenized = tokenizer.texts_to_sequences(new_data)
    to_pad = []
    for x in X_tokenized:
        if len(x) >0:
            to_pad.append(x[0])

    while len(to_pad) < 500:
        to_pad.append(0)

    test = [x for x in to_pad]
    test = [test]
    test = np.array(test)

    prediction = model_file.predict(test)
    pred_name = CATEGORIES[np.argmax(prediction)]
    return pred_name, prediction


new_data = "You should be fired, you're a moronic wimp who is too lazy to do research
predict_comment(new_data,tokenizer_toxic,toxic_lstm_single_10)
```

```
('toxic', array([[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
        0.16666667]], dtype=float32))
```

```
new_data = "I'm going to kill you!"
predict_comment(new_data,tokenizer_toxic,toxic_lstm_single_10)
```

```
('toxic', array([[0.16666667, 0.16666667, 0.16666667, 0.16666667, 0.16666667,
        0.16666667]], dtype=float32))
```

```
new_data = "You are a good person"
```

```
predict_comment(new_data,tokenizer_toxic,toxic_lstm_single_10)
```

```
('toxic', array([[0.16666667, 0.16666667, 0.16666667, 0.16666667,
        0.16666667]], dtype=float32))
```

```
ew_data = "After being kicked out of his rock band, Dewey Finn becomes a substitute t
redict_genre(new_data, tokenizer_movie, movie_lstm_single_10)
```

```
('Comedy', array([[0.38401654, 0.08035603, 0.0185371 , 0.13964197, 0.28417718,
        0.02190563, 0.6372429 , 0.16845113]], dtype=float32))
```

```
ew_data = "Amateur stuntman Rod Kimble has a problem--his step-father Frank is a jerk
redict_genre(new_data, tokenizer_movie, movie_lstm_single_10)
```

```
('Comedy', array([[0.10307938, 0.02601379, 0.02239436, 0.16088203, 0.0682241 ,
        0.01742652, 0.6710691 , 0.6104772 ]], dtype=float32))
```

```
#     test = np.array(to_pad)
#     test.reshape(1,200)
#     print(test.shape)
```