```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
import pickle
import numpy as np
import os



file = open("metamorphosis_clean (1).txt", "r", encoding = "utf8")
lines = []

for i in file:
    lines.append(i)

print("The First Line: ", lines[0])
print("The Last Line: ", lines[-1])
```

```
     The First Line:  One morning, when Gregor Samsa woke from troubled dreams, he found

     The Last Line:  first to get up and stretch out her young body.
```

```
data = ""

for i in lines:
    data = ' '. join(lines)

data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '')
data[:360]
```

```
     'One morning, when Gregor Samsa woke from troubled dreams, he found himself transfor
     med in his bed into a horrible vermin.  He lay on his armour-like back, and if he li
     fted his head a little he could see his brown belly, slightly domed and divided by a
     rches into stiff sections.  The bedding was hardly able to cover it and seemed ready
```

```
import string

translator = str.maketrans(string.punctuation, ' '*len(string.punctuation)) #map punctuation to space
new_data = data.translate(translator)

new_data[:500]
```

```
     'One morning  when Gregor Samsa woke from troubled dreams  he found himself transfor
     med in his bed into a horrible vermin   He lay on his armour like back  and if he li
     fted his head a little he could see his brown belly  slightly domed and divided by a
     rches into stiff sections   The bedding was hardly able to cover it and seemed ready
     to slide off any moment   His many legs  pitifully thin compared with the size of th
```

```
z = []

for i in data.split():
    if i not in z:
        z.append(i)

data = ' '.join(z)
data[:500]
```

```
     'One morning, when Gregor Samsa woke from troubled dreams, he found himself transfor
     med in his bed into a horrible vermin. He lay on armour-like back, and if lifted hea
     d little could see brown belly, slightly domed divided by arches stiff sections. The
     bedding was hardly able to cover it seemed ready slide off any moment. His many leg
     s, pitifully thin compared with the size of rest him, waved about helplessly as look
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([data])

pickle.dump(tokenizer, open('tokenizer1.pkl', 'wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]
sequence_data[:10]
```

```
     [17, 53, 293, 2, 18, 729, 135, 730, 294, 8]
```

```
vocab_size = len(tokenizer.word_index) + 1
print(vocab_size)
```

```
     2617
```

```
sequences = []

for i in range(1, len(sequence_data)):
    words = sequence_data[i-1:i+1]
    sequences.append(words)

print("The Length of sequences are: ", len(sequences))
sequences = np.array(sequences)
sequences[:10]
```

```
The Length of sequences are:  3889
array([[ 17,  53],
       [ 53, 293],
       [293,   2],
       [  2,  18],
       [ 18, 729],
       [729, 135],
       [135, 730],
       [730, 294],
       [294,   8],
       [  8, 731]])
```

```
X = []
y = []

for i in sequences:
    X.append(i[0])
    y.append(i[1])

X = np.array(X)
y = np.array(y)
```

```
print("The Data is: ", X[:5])
print("The responses are: ", y[:5])
```

```
The Data is:  [ 17  53 293   2  18]
The responses are:  [ 53 293   2  18 729]
```

```
print("Word Index:")
count = 0
for word, index in tokenizer.word_index.items():
    print(f"{word}: {index}", end=",\n" if count < 9 else "\n")
    count += 1
    if count >= 10:
        break
```

```
Word Index:
now: 1,
gregor: 2,
well: 3,
it: 4,
that: 5,
then: 6,
father: 7,
he: 8,
in: 9,
out: 10
```

```
print("Word Index:")
for word, index in tokenizer.word_index.items():
    print(f"{word}: {index}")
```

```
prevented: 2581
telling: 2582
peeved: 2583
cheerio: 2584
sharply: 2585
terribly: 2586
tonight: 2587
gets: 2588
destroyed: 2589
gained: 2590
twisted: 2591
stuff: 2592
kissed: 2593
hugged: 2594
country: 2595
sunshine: 2596
comfortably: 2597
seats: 2598
discussed: 2599
examination: 2600
jobs: 2601
promise: 2602
cheaper: 2603
location: 2604
practical: 2605
livelier: 2606
cheeks: 2607
pale: 2608
simultaneously: 2609
blossoming: 2610
built: 2611
quieter: 2612
other's: 2613
agreed: 2614
confirmation: 2615
destination: 2616
```

```
y = to_categorical(y, num_classes=vocab_size)
y[:5]
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

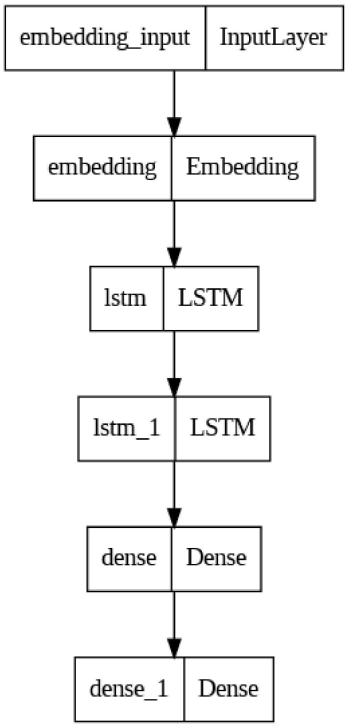
```
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=1))
model.add(LSTM(1000, return_sequences=True))
model.add(LSTM(1000))
model.add(Dense(1000, activation="relu"))
model.add(Dense(vocab_size, activation="softmax"))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1, 10)             26170

 lstm (LSTM)                 (None, 1, 1000)           4044000

 lstm_1 (LSTM)               (None, 1000)              8004000

 dense (Dense)               (None, 1000)              1001000

 dense_1 (Dense)             (None, 2617)              2619617

=================================================================
Total params: 15694787 (59.87 MB)
Trainable params: 15694787 (59.87 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from tensorflow import keras
from keras.utils import plot_model
print(plot_model)

keras.utils.plot_model(model, to_file='model.png', show_layer_names=True)
```

```
<function plot_model at 0x79212b3cf250>
```




```
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import TensorBoard

checkpoint = ModelCheckpoint("nextword1.h5", monitor='loss', verbose=1,
    save_best_only=True, mode='auto')

reduce = ReduceLROnPlateau(monitor='loss', factor=0.2, patience=3, min_lr=0.0001, verbose = 1)

logdir='logsnextword1'
tensorboard_Visualization = TensorBoard(log_dir=logdir)


model.compile(loss="categorical_crossentropy", optimizer=Adam(lr=0.001))
```

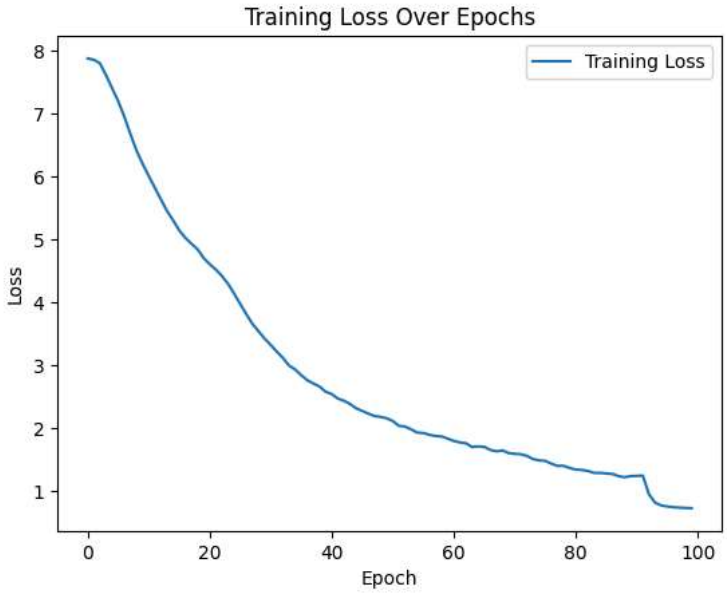
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers
```

```
model.fit(X, y, epochs=100, batch_size=64, callbacks=[checkpoint, reduce, tensorboard_Visualization])
```

```
Epoch 1/100
61/61 [==============================] - ETA: 0s - loss: 7.8754
Epoch 1: loss improved from inf to 7.87541, saving model to nextword1.h5
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 fil
  saving_api.save_model(
61/61 [==============================] - 25s 332ms/step - loss: 7.8754 - lr: 0.0010
Epoch 2/100
61/61 [==============================] - ETA: 0s - loss: 7.8549
Epoch 2: loss improved from 7.87541 to 7.85489, saving model to nextword1.h5
61/61 [==============================] - 20s 334ms/step - loss: 7.8549 - lr: 0.0010
Epoch 3/100
61/61 [==============================] - ETA: 0s - loss: 7.7997
Epoch 3: loss improved from 7.85489 to 7.79973, saving model to nextword1.h5
61/61 [==============================] - 20s 323ms/step - loss: 7.7997 - lr: 0.0010
Epoch 4/100
61/61 [==============================] - ETA: 0s - loss: 7.6113
Epoch 4: loss improved from 7.79973 to 7.61135, saving model to nextword1.h5
61/61 [==============================] - 30s 491ms/step - loss: 7.6113 - lr: 0.0010
Epoch 5/100
61/61 [==============================] - ETA: 0s - loss: 7.4052
Epoch 5: loss improved from 7.61135 to 7.40520, saving model to nextword1.h5
61/61 [==============================] - 20s 331ms/step - loss: 7.4052 - lr: 0.0010
Epoch 6/100
61/61 [==============================] - ETA: 0s - loss: 7.1965
Epoch 6: loss improved from 7.40520 to 7.19653, saving model to nextword1.h5
61/61 [==============================] - 21s 350ms/step - loss: 7.1965 - lr: 0.0010
Epoch 7/100
61/61 [==============================] - ETA: 0s - loss: 6.9535
Epoch 7: loss improved from 7.19653 to 6.95353, saving model to nextword1.h5
61/61 [==============================] - 20s 327ms/step - loss: 6.9535 - lr: 0.0010
Epoch 8/100
61/61 [==============================] - ETA: 0s - loss: 6.6769
Epoch 8: loss improved from 6.95353 to 6.67692, saving model to nextword1.h5
```

```
61/61 [==============================] - 21s 350ms/step - loss: 6.6769 - lr: 0.0010
Epoch 9/100
61/61 [==============================] - ETA: 0s - loss: 6.4145
Epoch 9: loss improved from 6.67692 to 6.41449, saving model to nextword1.h5
61/61 [==============================] - 20s 323ms/step - loss: 6.4145 - lr: 0.0010
Epoch 10/100
61/61 [==============================] - ETA: 0s - loss: 6.2051
Epoch 10: loss improved from 6.41449 to 6.20507, saving model to nextword1.h5
61/61 [==============================] - 21s 345ms/step - loss: 6.2051 - lr: 0.0010
Epoch 11/100
61/61 [==============================] - ETA: 0s - loss: 6.0122
Epoch 11: loss improved from 6.20507 to 6.01222, saving model to nextword1.h5
61/61 [==============================] - 20s 319ms/step - loss: 6.0122 - lr: 0.0010
Epoch 12/100
61/61 [==============================] - ETA: 0s - loss: 5.8240
Epoch 12: loss improved from 6.01222 to 5.82399, saving model to nextword1.h5
61/61 [==============================] - 20s 323ms/step - loss: 5.8240 - lr: 0.0010
Epoch 13/100
61/61 [==============================] - ETA: 0s - loss: 5.6374
Epoch 13: loss improved from 5.82399 to 5.63743, saving model to nextword1.h5
61/61 [==============================] - 19s 312ms/step - loss: 5.6374 - lr: 0.0010
Epoch 14/100
61/61 [==============================] - ETA: 0s - loss: 5.4556
Epoch 14: loss improved from 5.63743 to 5.45559, saving model to nextword1.h5
```

```
import matplotlib.pyplot as plt
# Retrieve the loss history from the model training
loss_history = model.history.history['loss']
# Plot the loss graph
plt.plot(loss_history, label='Training Loss')
plt.title('Training Loss Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



```
from tensorflow.keras.models import load_model
import numpy as np
import pickle

# Load the model and tokenizer

model = load_model('nextword1.h5')
tokenizer = pickle.load(open('tokenizer1.pkl', 'rb'))

def Predict_Next_Words(model, tokenizer, text):
    """
        In this function we are using the tokenizer and models trained
        and we are creating the sequence of the text entered and then
        using our model to predict and return the the predicted word.

    """
    for i in range(3):
        sequence = tokenizer.texts_to_sequences([text])[0]
        sequence = np.array(sequence)

        preds = model.predict_classes(sequence)
#         print(preds)
        predicted_word = ""

        for key, value in tokenizer.word_index.items():
            if value == preds:
                predicted_word = key
                break

        print(predicted_word)
        return predicted_word


from tensorflow.keras.models import load_model
import numpy as np
import pickle

# Load the model and tokenizer
model = load_model('nextword1.h5')
tokenizer = pickle.load(open('tokenizer1.pkl', 'rb'))

# Function to predict next words
def Predict_Next_Words(model, tokenizer, text):
    """
    In this function we are using the tokenizer and models trained
    and we are creating the sequence of the text entered and then
    using our model to predict and return the the predicted word.
    """
    sequence = tokenizer.texts_to_sequences([text])[0]
    sequence = np.array(sequence)
    preds = model.predict(np.expand_dims(sequence, axis=0))[0]

    # Get the index of the word with the highest probability
    pred_index = np.argmax(preds)

    # Map the index back to the word using the tokenizer
    predicted_word = tokenizer.index_word.get(pred_index, "Unknown")
    print(predicted_word)

    return predicted_word

# Main program
while True:
    text = input("Enter your line: ")

    if text == "stop the script":
        print("Ending The Program.....")
        break

    else:
        try:
            text = text.split(" ")
            text = text[-1]
            Predict_Next_Words(model, tokenizer, text)

        except Exception as e:
            print("An error occurred:", e)
            continue



 Enter your line: at the dull
 1/1 [==============================] - 1s 776ms/step
```

```
weather
Enter your line: collection of textile
```