```
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
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving Pride and Prejudice.txt to Pride and Prejudice (3).txt
```

```
file = open("Pride and Prejudice.txt", "r", encoding = "utf8")

# store file in list
lines = []
for i in file:
    lines.append(i)

# Convert list to string
data = ""
for i in lines:
  data = ' '. join(lines)

#replace unnecessary stuff with space
data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '').replace('“','').replace('”','')  #new line, carriage return, unicode ch

#remove unnecessary spaces
data = data.split()
data = ' '.join(data)
data[:500]
```

```
'The Project Gutenberg eBook of Pride and prejudice, by Jane Austen This eBook is
for the use of anyone anywhere in the United States and most other parts of the w
orld at no cost and with almost no restrictions whatsoever. You may copy it, give
it away or re-use it under the terms of the Project Gutenberg License included wi
th this eBook or online at www.gutenberg.org. If you are not located in the Unite
```

```
len(data)
```

```
733851
```

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

# saving the tokenizer for predict function
pickle.dump(tokenizer, open('token.pkl', 'wb'))

sequence_data = tokenizer.texts_to_sequences([data])[0]
sequence_data[:15]
```

```
[1, 182, 164, 1001, 3, 299, 4, 946, 30, 72, 710, 41, 1001, 23, 21]
```

```
len(sequence_data)
```

```
131237
```

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

```
7250
```

```
sequences = []

for i in range(3, len(sequence_data)):
    words = sequence_data[i-3:i+1]
    sequences.append(words)

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

```
The Length of sequences are:  131234
array([[   1,  182,  164, 1001],
       [ 182,  164, 1001,    3],
       [ 164, 1001,    3,  299],
       [1001,    3,  299,    4],
       [   3,  299,    4,  946],
       [ 299,    4,  946,   30],
       [   4,  946,   30,   72],
       [ 946,   30,   72,  710],
       [  30,   72,  710,   41],
       [  72,  710,   41, 1001]])
```

```
X = []
y = []

for i in sequences:
    X.append(i[0:3])
    y.append(i[3])

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

```
print("Data: ", X[:10])
print("Response: ", y[:10])
```

```
Data:  [[   1  182  164]
 [ 182  164 1001]
 [ 164 1001    3]
 [1001    3  299]
 [   3  299    4]
 [ 299    4  946]
 [   4  946   30]
 [ 946   30   72]
 [  30   72  710]
 [  72  710   41]]
Response:  [1001    3  299    4  946   30   72  710   41 1001]
```

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

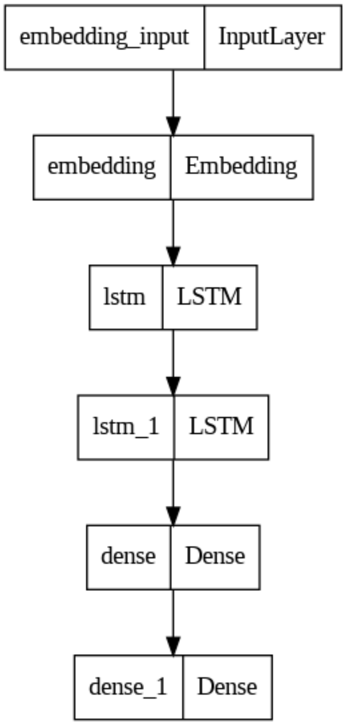
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

```
model = Sequential()
model.add(Embedding(vocab_size, 10, input_length=3))
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, 3, 10)             72500
 lstm (LSTM)                 (None, 3, 1000)           4044000

 lstm_1 (LSTM)               (None, 1000)              8004000

 dense (Dense)               (None, 1000)              1001000

 dense_1 (Dense)             (None, 7250)              7257250

=================================================================
Total params: 20,378,750
Trainable params: 20,378,750
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow import keras
from keras.utils.vis_utils import plot_model

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



```
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("next_words.h5", monitor='loss', verbose=1, save_best_only=True)
model.compile(loss="categorical_crossentropy", optimizer=Adam(learning_rate=0.001))
model.fit(X, y, epochs=70, batch_size=64, callbacks=[checkpoint])
```

```
    Epoch 37: loss improved from 0.63283 to 0.62363, saving model to next_words.h5
    2051/2051 [==============================] - 33s 16ms/step - loss: 0.6236
    Epoch 38/70
    2049/2051 [============================>.] - ETA: 0s - loss: 0.6088
    Epoch 38: loss improved from 0.62363 to 0.60894, saving model to next_words.h5
    2051/2051 [==============================] - 33s 16ms/step - loss: 0.6089
    Epoch 39/70
    2051/2051 [==============================] - ETA: 0s - loss: 0.6003
    Epoch 39: loss improved from 0.60894 to 0.60032, saving model to next_words.h5
    2051/2051 [==============================] - 33s 16ms/step - loss: 0.6003
    Epoch 40/70
    2049/2051 [============================>.] - ETA: 0s - loss: 0.5863
    Epoch 40: loss improved from 0.60032 to 0.58638, saving model to next_words.h5
    2051/2051 [==============================] - 36s 18ms/step - loss: 0.5864
    Epoch 41/70
    2050/2051 [============================>.] - ETA: 0s - loss: 0.5787
    Epoch 41: loss improved from 0.58638 to 0.57879, saving model to next_words.h5
    2051/2051 [==============================] - 32s 15ms/step - loss: 0.5788
    Epoch 42/70
    2051/2051 [==============================] - ETA: 0s - loss: 0.5727
    Epoch 42: loss improved from 0.57879 to 0.57266, saving model to next_words.h5
    2051/2051 [==============================] - 31s 15ms/step - loss: 0.5727
    Epoch 43/70
    2048/2051 [============================>.] - ETA: 0s - loss: 0.5607
    Epoch 43: loss improved from 0.57266 to 0.56090, saving model to next_words.h5
    2051/2051 [==============================] - 34s 16ms/step - loss: 0.5609
    Epoch 44/70
    2048/2051 [============================>.] - ETA: 0s - loss: 0.5555
    Epoch 44: loss improved from 0.56090 to 0.55557, saving model to next_words.h5
    2051/2051 [==============================] - 35s 17ms/step - loss: 0.5556
    Epoch 45/70
    2049/2051 [============================>.] - ETA: 0s - loss: 0.5483
    Epoch 45: loss improved from 0.55557 to 0.54833, saving model to next_words.h5
    2051/2051 [==============================] - 34s 16ms/step - loss: 0.5483
    Epoch 46/70
    2050/2051 [============================>.] - ETA: 0s - loss: 0.5410
    Epoch 46: loss improved from 0.54833 to 0.54105, saving model to next_words.h5
```

```
import joblib
filename = 'next_words.h5'
joblib.dump(model, filename)
```

```
    ['next_words.h5']
```

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

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

def Predict_Next_Words(model, tokenizer, text):

  sequence = tokenizer.texts_to_sequences([text])
  sequence = np.array(sequence)
  preds = np.argmax(model.predict(sequence))
  predicted_word = ""

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

  print(predicted_word)
  return predicted_word



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

  if text == "0":
      print("Execution completed.....")
      break

  else:
      try:
          text = text.split(" ")
          text = text[-3:]
          print(text)

          Predict_Next_Words(model, tokenizer, text)

      except Exception as e:
        print("Error occurred: ",e)
        continue
```

```
Enter your line: give it away
['give', 'it', 'away']
1/1 [==============================] - 1s 752ms/step
or
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-17-a124736db0da> in <cell line: 1>()
      1 while(True):
----> 2   text = input("Enter your line: ")
      3 
      4   if text == "0":
      5       print("Execution completed.....")
```

1 frames

```
/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in
_input_request(self, prompt, ident, parent, password)
    893             except KeyboardInterrupt:
    894                 # re-raise KeyboardInterrupt, to truncate traceback
--> 895                 raise KeyboardInterrupt("Interrupted by user") from None
    896             except Exception as e:
    897                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

SEARCH STACK OVERFLOW

```
%%writefile app.py
import streamlit as st
import pandas as pd
import numpy as np
import joblib

st.title('Next Word Predictor')

SO = st.number_input("Enter the word:")
loaded_model = joblib.load('next_words.h5')
inputs = (SO)
if st.button("Predict"):
    result = loaded_model.predict([inputs])
    st.write(result)
```

```
Overwriting app.py
```

```
!ngrok authtoken 2Kgk0JPvh53BRHvj2SCxcZI3YPN_2eDZG97kbtqGH7xy3yD88
```

```
Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

```
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
```

```
--2023-07-06 19:04:32--  https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
Resolving bin.equinox.io (bin.equinox.io)... 52.202.168.65, 54.237.133.81, 18.205.222.128, ...
Connecting to bin.equinox.io (bin.equinox.io)|52.202.168.65|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13921656 (13M) [application/octet-stream]
Saving to: ‘ngrok-stable-linux-amd64.zip.3’

ngrok-stable-linux- 100%[===================>]  13.28M  13.9MB/s    in 1.0s

2023-07-06 19:04:33 (13.9 MB/s) - ‘ngrok-stable-linux-amd64.zip.3’ saved [13921656/13921656]
```

```
!unzip ngrok-stable-linux-amd64.zip
```

```
    Archive:  ngrok-stable-linux-amd64.zip
    replace ngrok? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
      inflating: ngrok
```

```
get_ipython().system_raw('./ngrok http 8501 &')
```

```
! curl -s http://localhost:4040/api/tunnels | python3 -c \
"import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

```
    https://d2a7-34-90-196-79.ngrok-free.app
```

```
!streamlit run /content/app.py
```

```
    Collecting usage statistics. To deactivate, set browser.gatherUsageStats to False.


      You can now view your Streamlit app in your browser.

      Network URL: http://172.28.0.12:8501
      External URL: http://34.90.196.79:8501

      Stopping...
      Stopping...
```