Importing Libraries

```
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
```

Importing the data set

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

Choose Files Data Set.txt

- **Data Set.txt**(text/plain) - 204357 bytes, last modified: 5/2/2022 - 100% done

```
Saving Data Set.txt to Data Set (2).txt
```

Loading and Processing the data

```
file = open("Data Set.txt", "r", encoding = "utf8")

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

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

data = data.replace('\n', '').replace('\r', '').replace('\ufeff', '').replace('“','').replace

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

```
'The Project Gutenberg eBook of The Time Machine, by H. G. Wells This eBook is for the
use of anyone anywhere in the United States and most other parts of the world at no cos
t and with almost no restrictions whatsoever. You may copy it, give it away or re-use i
t under the terms of the Project Gutenberg License included with this eBook or online a
t www.gutenberg.org. If you are not located in the United States, you will have to chec
```

Checking the length of the data

```
len(data)
```

```
197921
```

Use of Tokenizer

```
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, 52, 44, 313, 3, 1, 18, 47, 31, 1159, 1160, 358, 21, 313, 32]
```

Removing the repeeated words count and white spaces and then counting the number of words

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

```
5163
```

Using words to predict next word

```
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:  35849
array([[   1,   52,   44,  313],
       [  52,   44,  313,    3],
       [  44,  313,    3,    1],
       [ 313,    3,    1,   18],
       [   3,    1,   18,   47],
       [   1,   18,   47,   31],
       [  18,   47,   31, 1159],
```

```
            [  47,   31, 1159, 1160],
            [  31, 1159, 1160,  358],
            [1159, 1160,  358,   21]])


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   52   44]
 [  52   44  313]
 [  44  313    3]
 [ 313    3    1]
 [   3    1   18]
 [   1   18   47]
 [  18   47   31]
 [  47   31 1159]
 [  31 1159 1160]
 [1159 1160  358]]
Response:  [ 313    3    1   18   47   31 1159 1160  358   21]
```

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

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

Model

```
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)             51630     

 lstm (LSTM)                 (None, 3, 1000)           4044000   

 lstm_1 (LSTM)               (None, 1000)              8004000   

 dense (Dense)               (None, 1000)              1001000   

 dense_1 (Dense)             (None, 5163)              5168163   

=================================================================
Total params: 18,268,793
Trainable params: 18,268,793
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)
```

embedding_input	InputLayer

Training the model

```
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 1/70
560/561 [============================>.] - ETA: 0s - loss: 6.6722
Epoch 1: loss improved from inf to 6.67232, saving model to next_words.h5
561/561 [==============================] - 27s 32ms/step - loss: 6.6723
Epoch 2/70
561/561 [==============================] - ETA: 0s - loss: 6.1925
Epoch 2: loss improved from 6.67232 to 6.19248, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 6.1925
Epoch 3/70
561/561 [==============================] - ETA: 0s - loss: 5.7929
Epoch 3: loss improved from 6.19248 to 5.79290, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 5.7929
Epoch 4/70
561/561 [==============================] - ETA: 0s - loss: 5.4677
Epoch 4: loss improved from 5.79290 to 5.46773, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 5.4677
Epoch 5/70
561/561 [==============================] - ETA: 0s - loss: 5.1998
Epoch 5: loss improved from 5.46773 to 5.19976, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 5.1998
Epoch 6/70
561/561 [==============================] - ETA: 0s - loss: 4.9565
Epoch 6: loss improved from 5.19976 to 4.95649, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 4.9565
Epoch 7/70
561/561 [==============================] - ETA: 0s - loss: 4.7365
Epoch 7: loss improved from 4.95649 to 4.73649, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 4.7365
Epoch 8/70
561/561 [==============================] - ETA: 0s - loss: 4.5210
Epoch 8: loss improved from 4.73649 to 4.52099, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 4.5210
Epoch 9/70
561/561 [==============================] - ETA: 0s - loss: 4.2851
Epoch 9: loss improved from 4.52099 to 4.28510, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 4.2851
Epoch 10/70
561/561 [==============================] - ETA: 0s - loss: 4.0289
Epoch 10: loss improved from 4.28510 to 4.02893, saving model to next_words.h5
561/561 [==============================] - 18s 31ms/step - loss: 4.0289
Epoch 11/70
561/561 [==============================] - ETA: 0s - loss: 3.7540
Epoch 11: loss improved from 4.02893 to 3.75401, saving model to next_words.h5
```

```
561/561 [==============================] - 18s 31ms/step - loss: 3.7540
Epoch 12/70
561/561 [==============================] - ETA: 0s - loss: 3.4626
Epoch 12: loss improved from 3.75401 to 3.46263, saving model to next_words.h5
561/561 [==============================] - 18s 32ms/step - loss: 3.4626
Epoch 13/70
561/561 [==============================] - ETA: 0s - loss: 3.1784
Epoch 13: loss improved from 3.46263 to 3.17843, saving model to next_words.h5
561/561 [==============================] - 18s 31ms/step - loss: 3.1784
Epoch 14/70
561/561 [==============================] - ETA: 0s - loss: 2.8978
Epoch 14: loss improved from 3.17843 to 2.89781, saving model to next_words.h5
561/561 [==============================] - 18s 31ms/step - loss: 2.8978
Epoch 15/70
561/561 [==============================] - ETA: 0s - loss: 2.6443
```

```
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)
```

```
        Predict_Next_Words(model, tokenizer, text)

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

```
Enter your line: the day today
['the', 'day', 'today']
i
Enter your line: for the event
['for', 'the', 'event']
time
Enter your line: random waords such as
['waords', 'such', 'as']
i
Enter your line: 0
Execution completed.....
```

✓ 49s completed at 10:13 PM