```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
from keras.layers import LSTM, Bidirectional
from keras.optimizers import Adam
```

```
token = Tokenizer(oov_token="$_$")
```

```
with open("dialogs.txt",'r') as da:
    data = da.read()
    text = data.lower().split( "\n")
texts =[]
ptext =[]
for items in text:
    ptext.append(items.split("\t"))
for item in ptext:
    texts.append(item[0])
    texts.append(item[1])
print(len(text))
print(texts)
```

```
3725
['hi, how are you doing?', "i'm fine. how about yourself?", "i'm fine. how about yourself?", "i'm pretty good. thanks for asking.",
```

```
token.fit_on_texts(texts)
print(token.word_index)
```

```
{'$_$': 1, 'i': 2, 'you': 3, 'the': 4, 'to': 5, 'a': 6, 'it': 7, 'that': 8, 'do': 9, 'what': 10, 'is': 11, 'of': 12, 'and': 13, 'ha
```

```
twords = len(token.word_index)+1
```

```
input_seq=[]
listofseq=[]
for line in texts:
    token_seq = token.texts_to_sequences([line])[0]
    listofseq.append(token_seq)
    length=len(token_seq)
real_seq = list(filter(None, listofseq))
for seq in real_seq:
    size = len(seq)
    for i in range(1,size):
        input_seq.append(seq[:i+1])
```

```
max_length = max([len(x) for x in real_seq])
print(len(input_seq))
```

```
40643
```

```
input_seq = np.array(pad_sequences(input_seq,maxlen= max_length,padding ='pre'))
print(input_seq[0])
```

```
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0 1523   37]
```

```
train = input_seq[:,:-1]
label = input_seq[:,-1]
y = tf.keras.utils.to_categorical(label,num_classes=twords)
```

```
print(y.shape)
```

```
(40643, 2521)
```

```
# bhai plz is ko barbar run mat ker new model bante hai
model1 = keras.Sequential([keras.layers.Embedding(twords,240,input_length = max_length-1),
                           keras.layers.Bidirectional(LSTM(150)),
                           keras.layers.Dense(twords, activation='softmax') ])
model1.summary()
adam =Adam(learning_rate =0.1)
model1.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 18, 240)           605040

 bidirectional (Bidirection  (None, 300)               469200
 al)

 dense (Dense)               (None, 2521)              758821

=================================================================
Total params: 1833061 (6.99 MB)
Trainable params: 1833061 (6.99 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# final_model = tf.keras.models.load_model('chatgen1.keras')
```

```
history = model1.fit(train[:10000],y[:10000],epochs =50,verbose =1)
```

```
Epoch 22/50
313/313 [==============================] - 2s 8ms/step - loss: 14.8566 - accuracy: 0.2094
Epoch 23/50
313/313 [==============================] - 4s 13ms/step - loss: 14.2050 - accuracy: 0.2355
Epoch 24/50
313/313 [==============================] - 3s 9ms/step - loss: 13.7717 - accuracy: 0.2494
Epoch 25/50
313/313 [==============================] - 3s 10ms/step - loss: 14.3865 - accuracy: 0.2434
Epoch 26/50
313/313 [==============================] - 3s 10ms/step - loss: 13.3278 - accuracy: 0.2617
Epoch 27/50
313/313 [==============================] - 4s 13ms/step - loss: 13.4146 - accuracy: 0.2585
Epoch 28/50
```

```
313/313 [==============================] - 3s 10ms/step - loss: 12.5146 - accuracy: 0.3463
Epoch 50/50
313/313 [==============================] - 3s 9ms/step - loss: 12.2934 - accuracy: 0.3482
```

```
model1.fit(train[10000:20000],y[10000:20000],epochs =50,verbose =1)
```

```
313/313 [==============================] - 3s 9ms/step - loss: 10.5042 - accuracy: 0.4839
Epoch 23/50
313/313 [==============================] - 3s 9ms/step - loss: 10.9373 - accuracy: 0.4793
Epoch 24/50
313/313 [==============================] - 2s 8ms/step - loss: 10.8625 - accuracy: 0.4817
Epoch 25/50
313/313 [==============================] - 3s 8ms/step - loss: 11.1565 - accuracy: 0.4723
Epoch 26/50
313/313 [==============================] - 3s 8ms/step - loss: 11.2858 - accuracy: 0.4795
Epoch 27/50
313/313 [==============================] - 3s 11ms/step - loss: 12.0950 - accuracy: 0.4595
Epoch 28/50
313/313 [==============================] - 3s 8ms/step - loss: 11.1688 - accuracy: 0.4689
Epoch 29/50
313/313 [==============================] - 2s 8ms/step - loss: 10.5223 - accuracy: 0.4843
Epoch 30/50
313/313 [==============================] - 3s 8ms/step - loss: 10.8691 - accuracy: 0.4778
Epoch 31/50
313/313 [==============================] - 3s 8ms/step - loss: 10.3945 - accuracy: 0.5014
Epoch 32/50
313/313 [==============================] - 3s 10ms/step - loss: 10.3897 - accuracy: 0.4879
Epoch 33/50
313/313 [==============================] - 2s 8ms/step - loss: 10.3344 - accuracy: 0.4925
Epoch 34/50
313/313 [==============================] - 3s 8ms/step - loss: 10.3147 - accuracy: 0.4946
Epoch 35/50
313/313 [==============================] - 3s 8ms/step - loss: 10.2297 - accuracy: 0.4974
Epoch 36/50
313/313 [==============================] - 3s 10ms/step - loss: 10.4518 - accuracy: 0.4956
Epoch 37/50
313/313 [==============================] - 3s 9ms/step - loss: 10.8036 - accuracy: 0.4906
Epoch 38/50
313/313 [==============================] - 3s 8ms/step - loss: 10.3691 - accuracy: 0.4981
Epoch 39/50
313/313 [==============================] - 2s 8ms/step - loss: 10.8170 - accuracy: 0.4881
Epoch 40/50
313/313 [==============================] - 2s 8ms/step - loss: 10.1037 - accuracy: 0.5011
Epoch 41/50
313/313 [==============================] - 3s 11ms/step - loss: 10.2146 - accuracy: 0.5044
Epoch 42/50
313/313 [==============================] - 3s 8ms/step - loss: 10.8293 - accuracy: 0.4900
Epoch 43/50
313/313 [==============================] - 2s 8ms/step - loss: 10.6793 - accuracy: 0.4957
Epoch 44/50
313/313 [==============================] - 2s 8ms/step - loss: 10.9510 - accuracy: 0.4902
Epoch 45/50
313/313 [==============================] - 3s 8ms/step - loss: 10.7325 - accuracy: 0.4906
Epoch 46/50
313/313 [==============================] - 3s 10ms/step - loss: 12.1945 - accuracy: 0.4709
Epoch 47/50
313/313 [==============================] - 2s 8ms/step - loss: 12.7968 - accuracy: 0.4498
Epoch 48/50
313/313 [==============================] - 3s 9ms/step - loss: 11.8100 - accuracy: 0.4648
Epoch 49/50
313/313 [==============================] - 2s 8ms/step - loss: 11.6000 - accuracy: 0.4751
Epoch 50/50
313/313 [==============================] - 3s 9ms/step - loss: 11.4575 - accuracy: 0.4791
<keras.src.callbacks.History at 0x7d2b765cf6a0>
```

```
model1.fit(train[20000:30000],y[20000:30000],epochs =50,verbose =1)
```

```
313/313 [==============================] - 2s 8ms/step - loss: 10.7448 - accuracy: 0.4867
Epoch 33/50
313/313 [==============================] - 2s 8ms/step - loss: 10.7554 - accuracy: 0.4860
Epoch 34/50
313/313 [==============================] - 2s 8ms/step - loss: 10.4626 - accuracy: 0.4930
Epoch 35/50
313/313 [==============================] - 3s 9ms/step - loss: 10.5997 - accuracy: 0.4971
Epoch 36/50
313/313 [==============================] - 3s 9ms/step - loss: 9.3871 - accuracy: 0.5235
Epoch 37/50
313/313 [==============================] - 2s 8ms/step - loss: 9.7240 - accuracy: 0.5257
Epoch 38/50
313/313 [==============================] - 2s 8ms/step - loss: 9.4363 - accuracy: 0.5202
Epoch 39/50
313/313 [==============================] - 3s 8ms/step - loss: 10.0729 - accuracy: 0.5066
Epoch 40/50
313/313 [==============================] - 3s 10ms/step - loss: 10.2861 - accuracy: 0.5103
Epoch 41/50
313/313 [==============================] - 3s 8ms/step - loss: 9.6577 - accuracy: 0.5223
Epoch 42/50
313/313 [==============================] - 2s 8ms/step - loss: 9.3917 - accuracy: 0.5290
Epoch 43/50
313/313 [==============================] - 2s 8ms/step - loss: 9.9494 - accuracy: 0.5206
Epoch 44/50
313/313 [==============================] - 2s 8ms/step - loss: 9.6874 - accuracy: 0.5245
Epoch 45/50
313/313 [==============================] - 3s 10ms/step - loss: 9.4136 - accuracy: 0.5371
Epoch 46/50
313/313 [==============================] - 2s 7ms/step - loss: 9.8123 - accuracy: 0.5216
Epoch 47/50
313/313 [==============================] - 2s 8ms/step - loss: 10.0286 - accuracy: 0.5245
Epoch 48/50
313/313 [==============================] - 2s 7ms/step - loss: 10.6076 - accuracy: 0.5072
Epoch 49/50
313/313 [==============================] - 2s 7ms/step - loss: 9.9027 - accuracy: 0.5271
Epoch 50/50
313/313 [==============================] - 3s 10ms/step - loss: 10.2528 - accuracy: 0.5211
<keras.src.callbacks.History at 0x7d2b77f18be0>
```

```
model1.fit(train[30000:],y[30000:],epochs =50,verbose =1)
```

```
313/313 [==============================] - 3s 11ms/step - loss: 12.0007 - accuracy: 0.3877
Epoch 23/50
313/313 [==============================] - 2s 8ms/step - loss: 11.3009 - accuracy: 0.4141
Epoch 24/50
313/313 [==============================] - 3s 9ms/step - loss: 11.0520 - accuracy: 0.4153
```

```
313/313 [==============================] - 2s 8ms/step - loss: 10.6996 - accuracy: 0.4564
Epoch 47/50
313/313 [==============================] - 2s 8ms/step - loss: 11.1128 - accuracy: 0.4463
Epoch 48/50
313/313 [==============================] - 2s 8ms/step - loss: 10.9410 - accuracy: 0.4408
Epoch 49/50
313/313 [==============================] - 3s 8ms/step - loss: 12.0291 - accuracy: 0.4338
Epoch 50/50
313/313 [==============================] - 3s 10ms/step - loss: 12.3608 - accuracy: 0.4208
<keras.src.callbacks.History at 0x7d2b76f55ba0>
```

```
model1.save("dialogbot.keras")
```

```
print(type(token.word_index))
print((token.word_index))
def get_key(val):
   for key, value in token.word_index.items():
        if val == value:
            return key
```

```
<class 'dict'>
{'$_$': 1, 'i': 2, 'you': 3, 'the': 4, 'to': 5, 'a': 6, 'it': 7, 'that': 8, 'do': 9, 'what': 10, 'is': 11, 'of': 12, 'and': 13, 'ha
```

```
seed_text = "i am "
next_words = 100
for _ in range(next_words):
  token_list = token.texts_to_sequences([seed_text])[0]
  token_list = pad_sequences([token_list], maxlen=max_length-1, padding='pre')
  predicted = model1.predict(token_list, verbose=0)
  # print(max(predicted[0]))
  a = max(predicted[0])
  predicted=predicted.tolist()
  b = predicted[0].index(a)
  # print(b)
  output_word = get_key(b)
  seed_text += " "+ output_word

print(seed_text)
with open("output2.txt",'w') as f:
    f.write(seed_text)
```

```
i spend tonight student getting full 30 000 using young anymore hard then water into fat accident poetry friends needs cars yet guys
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.