In [1]:

```
!gdown --id 1OurDQUtbWQacvT32HMqFL7vIUrSMllOp
```

```
Downloading...
From: https://drive.google.com/uc?id=1OurDQUtbWQacvT32HMqFL7vIUrSMllOp
To: /content/preprocessed_data.csv
100% 300k/300k [00:00<00:00, 43.8MB/s]
```

In [62]:

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2
.8.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.41.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.5.30
)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (5.
0.2)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kag
gle) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests
->kaggle) (3.0.4)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python
-slugify->kaggle) (1.3)
```

In [63]:

```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 /root/.kaggle/kaggle.json
!kaggle datasets download -d yekenot/fasttext-crawl-300d-2m
```

```
Downloading fasttext-crawl-300d-2m.zip to /content
 99% 1.42G/1.44G [00:22<00:00, 111MB/s]
100% 1.44G/1.44G [00:22<00:00, 68.4MB/s]
```

In [64]:

```
!7z e fasttext-crawl-300d-2m.zip -o/content -r
```

```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Xeon(R) CPU @
2.20GHz (406F0),ASM,AES-NI)

Scanning the drive for archives:
  0M Scan                           1 file, 1545551987 bytes (1474 MiB)

Extracting archive: fasttext-crawl-300d-2m.zip
--
Path = fasttext-crawl-300d-2m.zip
Type = zip
Physical Size = 1545551987

  0%              0% - crawl-300d-2M.vec
                           1% - crawl-300d-2M.vec
                  2% - crawl-300d-2M.vec
                           3% - crawl-300d-2M.vec
                  4% - crawl-300d-2M.vec
                           5% - crawl-300d-2M.vec
                  6% - crawl-300d-2M.vec
                           7% - crawl-300d-2M.vec
                  8% - crawl-300d-2M.vec
                           9% - crawl-300d-2M.vec
                 10% - crawl-300d-2M.vec
                          11% - crawl-300d-2M.vec
                          12% - crawl-300d-2M.vec
                          13% - crawl-300d-2M.vec
                          14% - crawl-300d-2M.vec
                          15% - crawl-300d-2M.vec
                          16% - crawl-300d-2M.vec
                          17% - crawl-300d-2M.vec
                          18% - crawl-300d-2M.vec
                          19% - crawl-300d-2M.vec
                          20% - crawl-300d-2M.vec
                          21% - crawl-300d-2M.vec
                          22% - crawl-300d-2M.vec
                          23% - crawl-300d-2M.vec
```

 24% - crawl-300d-2M.vec
 25% - crawl-300d-2M.vec
 26% - crawl-300d-2M.vec
 27% - crawl-300d-2M.vec
 28% - crawl-300d-2M.vec
 29% - crawl-300d-2M.vec
 30% - crawl-300d-2M.vec
 31% - crawl-300d-2M.vec
 32% - crawl-300d-2M.vec
 33% - crawl-300d-2M.vec
 34% - crawl-300d-2M.vec
 35% - crawl-300d-2M.vec
 36% - crawl-300d-2M.vec
 37% - crawl-300d-2M.vec
 38% - crawl-300d-2M.vec
 39% - crawl-300d-2M.vec
 40% - crawl-300d-2M.vec
 41% - crawl-300d-2M.vec
 42% - crawl-300d-2M.vec
 43% - crawl-300d-2M.vec
 44% - crawl-300d-2M.vec
 45% - crawl-300d-2M.vec
 46% - crawl-300d-2M.vec
 47% - crawl-300d-2M.vec
 48% - crawl-300d-2M.vec
 49% - crawl-300d-2M.vec
 50% - crawl-300d-2M.vec
 51% - crawl-300d-2M.vec
 52% - crawl-300d-2M.vec
 53% - crawl-300d-2M.vec
 54% - crawl-300d-2M.vec
 55% - crawl-300d-2M.vec
 56% - crawl-300d-2M.vec
 57% - crawl-300d-2M.vec
 58% - crawl-300d-2M.vec
 59% - crawl-300d-2M.vec
 60% - crawl-300d-2M.vec
 61% - crawl-300d-2M.vec
 62% - crawl-300d-2M.vec
 63% - crawl-300d-2M.vec
 64% - crawl-300d-2M.vec
 65% - crawl-300d-2M.vec
 66% - crawl-300d-2M.vec
 67% - crawl-300d-2M.vec
 68% - crawl-300d-2M.vec
 69% - crawl-300d-2M.vec
 70% - crawl-300d-2M.vec
 71% - crawl-300d-2M.vec
 72% - crawl-300d-2M.vec
 73% - crawl-300d-2M.vec
 74% - crawl-300d-2M.vec
 75% - crawl-300d-2M.vec
 76% - crawl-300d-2M.vec
 77% - crawl-300d-2M.vec
 78% - crawl-300d-2M.vec
 79% - crawl-300d-2M.vec
 80% - crawl-300d-2M.vec
 81% - crawl-300d-2M.vec
 82% - crawl-300d-2M.vec
 83% - crawl-300d-2M.vec
 84% - crawl-300d-2M.vec
 85% - crawl-300d-2M.vec
 86% - crawl-300d-2M.vec
 87% - crawl-300d-2M.vec
 88% - crawl-300d-2M.vec
 89% - crawl-300d-2M.vec
 90% - crawl-300d-2M.vec
 91% - crawl-300d-2M.vec
 92% - crawl-300d-2M.vec
 93% - crawl-300d-2M.vec
 94% - crawl-300d-2M.vec
 95% - crawl-300d-2M.vec
 96% - crawl-300d-2M.vec
 97% - crawl-300d-2M.vec
 98% - crawl-300d-2M.vec
 99% - crawl-300d-2M.vec
100% - crawl-300d-2M.vec

```
                    Everything is Ok

Size:       4516698366
Compressed: 1545551987
```

In [65]:

```
# Reading fast text vectors in python: https://stackoverflow.com/a/38230349/4084039
def fasttextModel(gloveFile):
    print ("Loading Fasttext Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}#for storing word and the corresponding embedding vector for that word
    for line in f:
        splitLine = line.split()#splitting the line and storing it in a list
        word = splitLine[0]#getting the first element and storing it in word
        embedding = np.array([float(val) for val in splitLine[1:]])#obtaining corresponding vector for th
        model[word] = embedding#storing word as key and embedding vector for that word as value
    print ("Done.",len(model)," words loaded!")
    return model
model = fasttextModel('/content/crawl-300d-2M.vec')
```

```
Loading Fasttext Model
Done. 2000000  words loaded!
```

In [2]:

```
#Importing necessary libraries
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

In [3]:

```
df=pd.read_csv('preprocessed_data.csv')#reading data into DataFrame
```

In [4]:

```
df.head(4)#displaying top 4 datapoints
```

Out[4]:

	Unnamed: 0	source	target
0	0	U wan me to "chop" seat 4 u nt?\n	Do you want me to reserve seat for you or not?\n
1	1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	3	I'm thai. what do u do?\n	I'm Thai. What do you do?\n

In [5]:

```
def preprocess(x):#removing last character
  x=x[:-1]
  return x
```

In [6]:

```
df['source']=df['source'].apply(preprocess)#preprocessing source data
df['target']=df['target'].apply(preprocess)#preprocessing target data
```

In [7]:

```
df=df[['source','target']]
df.head()
```

Out[7]:

	source	target
0	U wan me to "chop" seat 4 u nt?	Do you want me to reserve seat for you or not?
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	I'm thai. what do u do?	I'm Thai. What do you do?
4	Hi! How did your week go? Haven heard from you...	Hi! How did your week go? Haven't heard from y...

In [8]:

```
df.shape#shape of DataFrame
```

Out[8]:

```
(2000, 2)
```

In [9]:

```
df=df[df['source'].apply(len)<170]#removing source sentences of length greater than or equal to 170
df=df[df['target'].apply(len)<200]#removing target sentences of length greater than or equal to 200
```

In [10]:

```
df.shape#shape of DataFrame
```

Out[10]:

```
(1990, 2)
```

In [11]:

```
from sklearn.model_selection import train_test_split
X=df['source']
y=df['target']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.01)#splitting the data in the ratio 99:1
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1970,)
(20,)
(1970,)
(20,)
```

In [57]:

```
X_train.to_csv('X_train.csv')
y_train.to_csv('y_train.csv')
X_test.to_csv('X_test.csv')
y_test.to_csv('y_test.csv')
```

Target:

In [13]:

```
target_tokenizer= Tokenizer()#tokenization on target
target_tokenizer.fit_on_texts(y_train)#fitting on ytrain
target_vocab_size= len(target_tokenizer.word_index) + 1#target vocab size
print(len(target_tokenizer.word_index))
```

```
3032
```

In [14]:

```
target_encoded_docs_train = target_tokenizer.texts_to_sequences(y_train)#converting text to integers
target_encoded_docs_test = target_tokenizer.texts_to_sequences(y_test)#converting text to integers
```

In [15]:

```
target_padded_docs_train = pad_sequences(target_encoded_docs_train,padding='post')#padding to maxlength
```

In [16]:

```
target_padded_docs_train.shape
```

Out[16]:

```
(1970, 43)
```

In [17]:

```
target_padded_docs_test = pad_sequences(target_encoded_docs_test,maxlen=target_padded_docs_train.shape[1]
```

In [18]:

```
target_padded_docs_test.shape
```

Out[18]:

```
(20, 43)
```

Source:

In [19]:

```
source_tokenizer= Tokenizer()#tokenization on source
source_tokenizer.fit_on_texts(X_train)#fitting to X_train
source_vocab_size= len(source_tokenizer.word_index) + 1#source vocab size
print(len(source_tokenizer.word_index))
```

```
3703
```

In [20]:

```
source_encoded_docs_train = source_tokenizer.texts_to_sequences(X_train)#converting text to sequence
source_encoded_docs_test = source_tokenizer.texts_to_sequences(X_test)#converting text to sequence
```

In [21]:

```
source_padded_docs_train = pad_sequences(source_encoded_docs_train,maxlen=target_padded_docs_train.shape[
```

In [22]:

```
source_padded_docs_train.shape
```

Out[22]:

```
(1970, 43)
```

In [23]:

```
source_padded_docs_test = pad_sequences(source_encoded_docs_test,maxlen=target_padded_docs_train.shape[1]
```

In [24]:

```
source_padded_docs_test.shape
```

Out[24]:

```
(20, 43)
```

In [25]:

```
#we are reshaping the dataset because the sparese_categorical_crossentropy requires data to be three dim

target_padded_docs_train=target_padded_docs_train.reshape((*target_padded_docs_train.shape,1))
target_padded_docs_test=target_padded_docs_test.reshape((*target_padded_docs_test.shape,1))
```

In [26]:

```
print(target_padded_docs_train.shape)
print(target_padded_docs_test.shape)
```

```
(1970, 43, 1)
(20, 43, 1)
```

In [27]:

```
#we are reshaping the dataset because the sparese_categorical_crossentropy requires data to be three dim

source_padded_docs_train=source_padded_docs_train.reshape((*source_padded_docs_train.shape,1))
source_padded_docs_test=source_padded_docs_test.reshape((*source_padded_docs_test.shape,1))
```

In [28]:

```
print(source_padded_docs_train.shape)
print(source_padded_docs_test.shape)
```

```
(1970, 43, 1)
(20, 43, 1)
```

In [59]:

```
import pandas as pd
pd.DataFrame(source_encoded_docs_train).to_csv("source_encoded_docs_train.csv")
pd.DataFrame(source_encoded_docs_test).to_csv("source_encoded_docs_test.csv")
pd.DataFrame(target_encoded_docs_train).to_csv("target_encoded_docs_train.csv")
pd.DataFrame(target_encoded_docs_test).to_csv("target_encoded_docs_test.csv")
```

In [69]:

```
#creating embedding matrix
embedding_matrix = np.zeros((source_vocab_size, 300))
for word, i in source_tokenizer.word_index.items():
    embedding_vector = model.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [70]:

```
embedding_matrix.shape
```

Out[70]:

```
(3704, 300)
```

Model1:

In [75]:

```
input=tf.keras.layers.Input(shape=(43,))
embed=tf.keras.layers.Embedding(source_vocab_size,300,weights=[embedding_matrix],input_length=source_padc
lstm1=tf.keras.layers.LSTM(128, return_sequences=True)(embed)
output=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(target_vocab_size, activation='softmax'))(ls
model=tf.keras.models.Model(inputs=input,outputs=output)
model.summary()
```

```
Model: "model_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 43)]              0
_________________________________________________________________
embedding_5 (Embedding)      (None, 43, 300)           1111200
_________________________________________________________________
lstm_5 (LSTM)                (None, 43, 128)           219648
_________________________________________________________________
time_distributed_9 (TimeDist (None, 43, 3033)          391257
=================================================================
Total params: 1,722,105
Trainable params: 610,905
Non-trainable params: 1,111,200
_________________________________________________________________
```

In [76]:

```
# Compile model
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [77]:

```
model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=50,
          validation_data=(source_padded_docs_test,target_padded_docs_test))
```

```
Epoch 1/50
2/2 [==============================] - 14s 542ms/step - loss: 7.9000 - accuracy: 0.3258 - val_loss:
6.3978 - val_accuracy: 0.6814
Epoch 2/50
2/2 [==============================] - 0s 193ms/step - loss: 5.9089 - accuracy: 0.6740 - val_loss:
4.0366 - val_accuracy: 0.6814
Epoch 3/50
2/2 [==============================] - 0s 191ms/step - loss: 3.5780 - accuracy: 0.6740 - val_loss:
2.5258 - val_accuracy: 0.6814
Epoch 4/50
2/2 [==============================] - 0s 191ms/step - loss: 2.7380 - accuracy: 0.6737 - val_loss:
2.9426 - val_accuracy: 0.6814
Epoch 5/50
2/2 [==============================] - 0s 193ms/step - loss: 3.0946 - accuracy: 0.6737 - val_loss:
2.9639 - val_accuracy: 0.6814
Epoch 6/50
2/2 [==============================] - 0s 192ms/step - loss: 3.0431 - accuracy: 0.6737 - val_loss:
2.7388 - val_accuracy: 0.6814
Epoch 7/50
2/2 [==============================] - 0s 196ms/step - loss: 2.7995 - accuracy: 0.6737 - val_loss:
2.5578 - val_accuracy: 0.6814
Epoch 8/50
2/2 [==============================] - 0s 195ms/step - loss: 2.6768 - accuracy: 0.6740 - val_loss:
2.5907 - val_accuracy: 0.6837
Epoch 9/50
2/2 [==============================] - 0s 196ms/step - loss: 2.7035 - accuracy: 0.6759 - val_loss:
2.5467 - val_accuracy: 0.6837
Epoch 10/50
2/2 [==============================] - 0s 193ms/step - loss: 2.6301 - accuracy: 0.6757 - val_loss:
2.4369 - val_accuracy: 0.6837
Epoch 11/50
2/2 [==============================] - 0s 192ms/step - loss: 2.5237 - accuracy: 0.6756 - val_loss:
2.3866 - val_accuracy: 0.6872
Epoch 12/50
2/2 [==============================] - 0s 196ms/step - loss: 2.4805 - accuracy: 0.6766 - val_loss:
2.3524 - val_accuracy: 0.6872
Epoch 13/50
2/2 [==============================] - 0s 197ms/step - loss: 2.4363 - accuracy: 0.6766 - val_loss:
2.2853 - val_accuracy: 0.6872
Epoch 14/50
2/2 [==============================] - 0s 195ms/step - loss: 2.3663 - accuracy: 0.6768 - val_loss:
2.2185 - val_accuracy: 0.6884
Epoch 15/50
2/2 [==============================] - 0s 196ms/step - loss: 2.3002 - accuracy: 0.6784 - val_loss:
2.1702 - val_accuracy: 0.6884
Epoch 16/50
2/2 [==============================] - 0s 197ms/step - loss: 2.2507 - accuracy: 0.6789 - val_loss:
2.1286 - val_accuracy: 0.6907
Epoch 17/50
2/2 [==============================] - 0s 195ms/step - loss: 2.2092 - accuracy: 0.6787 - val_loss:
2.0997 - val_accuracy: 0.6884
Epoch 18/50
2/2 [==============================] - 0s 193ms/step - loss: 2.1720 - accuracy: 0.6786 - val_loss:
2.0669 - val_accuracy: 0.6895
Epoch 19/50
2/2 [==============================] - 0s 195ms/step - loss: 2.1344 - accuracy: 0.6805 - val_loss:
2.0358 - val_accuracy: 0.6919
Epoch 20/50
2/2 [==============================] - 0s 198ms/step - loss: 2.1012 - accuracy: 0.6821 - val_loss:
2.0027 - val_accuracy: 0.6919
Epoch 21/50
2/2 [==============================] - 0s 194ms/step - loss: 2.0770 - accuracy: 0.6873 - val_loss:
1.9834 - val_accuracy: 0.6942
Epoch 22/50
2/2 [==============================] - 0s 195ms/step - loss: 2.0548 - accuracy: 0.6903 - val_loss:
1.9644 - val_accuracy: 0.6919
Epoch 23/50
2/2 [==============================] - 0s 200ms/step - loss: 2.0354 - accuracy: 0.6911 - val_loss:
1.9430 - val_accuracy: 0.6942
Epoch 24/50
2/2 [==============================] - 0s 197ms/step - loss: 2.0174 - accuracy: 0.6918 - val_loss:
```

1.9276 - val_accuracy: 0.6953
Epoch 25/50
2/2 [==============================] - 0s 195ms/step - loss: 2.0004 - accuracy: 0.6918 - val_loss:
1.9112 - val_accuracy: 0.6930
Epoch 26/50
2/2 [==============================] - 0s 195ms/step - loss: 1.9818 - accuracy: 0.6919 - val_loss:
1.8940 - val_accuracy: 0.6942
Epoch 27/50
2/2 [==============================] - 0s 194ms/step - loss: 1.9616 - accuracy: 0.6928 - val_loss:
1.8747 - val_accuracy: 0.6977
Epoch 28/50
2/2 [==============================] - 0s 195ms/step - loss: 1.9408 - accuracy: 0.6949 - val_loss:
1.8585 - val_accuracy: 0.7023
Epoch 29/50
2/2 [==============================] - 0s 195ms/step - loss: 1.9196 - accuracy: 0.6978 - val_loss:
1.8417 - val_accuracy: 0.7035
Epoch 30/50
2/2 [==============================] - 0s 196ms/step - loss: 1.8981 - accuracy: 0.6991 - val_loss:
1.8260 - val_accuracy: 0.7047
Epoch 31/50
2/2 [==============================] - 0s 195ms/step - loss: 1.8762 - accuracy: 0.7004 - val_loss:
1.8115 - val_accuracy: 0.7058
Epoch 32/50
2/2 [==============================] - 0s 195ms/step - loss: 1.8564 - accuracy: 0.7016 - val_loss:
1.7919 - val_accuracy: 0.7070
Epoch 33/50
2/2 [==============================] - 0s 202ms/step - loss: 1.8339 - accuracy: 0.7040 - val_loss:
1.7763 - val_accuracy: 0.7128
Epoch 34/50
2/2 [==============================] - 0s 196ms/step - loss: 1.8128 - accuracy: 0.7064 - val_loss:
1.7558 - val_accuracy: 0.7105
Epoch 35/50
2/2 [==============================] - 0s 197ms/step - loss: 1.7905 - accuracy: 0.7083 - val_loss:
1.7424 - val_accuracy: 0.7174
Epoch 36/50
2/2 [==============================] - 0s 196ms/step - loss: 1.7671 - accuracy: 0.7118 - val_loss:
1.7254 - val_accuracy: 0.7221
Epoch 37/50
2/2 [==============================] - 0s 196ms/step - loss: 1.7449 - accuracy: 0.7155 - val_loss:
1.7112 - val_accuracy: 0.7279
Epoch 38/50
2/2 [==============================] - 0s 198ms/step - loss: 1.7216 - accuracy: 0.7208 - val_loss:
1.6903 - val_accuracy: 0.7267
Epoch 39/50
2/2 [==============================] - 0s 200ms/step - loss: 1.6983 - accuracy: 0.7233 - val_loss:
1.6748 - val_accuracy: 0.7302
Epoch 40/50
2/2 [==============================] - 0s 198ms/step - loss: 1.6742 - accuracy: 0.7264 - val_loss:
1.6576 - val_accuracy: 0.7314
Epoch 41/50
2/2 [==============================] - 0s 197ms/step - loss: 1.6527 - accuracy: 0.7293 - val_loss:
1.6454 - val_accuracy: 0.7372
Epoch 42/50
2/2 [==============================] - 0s 209ms/step - loss: 1.6293 - accuracy: 0.7323 - val_loss:
1.6332 - val_accuracy: 0.7395
Epoch 43/50
2/2 [==============================] - 0s 198ms/step - loss: 1.6061 - accuracy: 0.7346 - val_loss:
1.6162 - val_accuracy: 0.7407
Epoch 44/50
2/2 [==============================] - 0s 200ms/step - loss: 1.5830 - accuracy: 0.7381 - val_loss:
1.5977 - val_accuracy: 0.7407
Epoch 45/50
2/2 [==============================] - 0s 196ms/step - loss: 1.5606 - accuracy: 0.7406 - val_loss:
1.5836 - val_accuracy: 0.7488
Epoch 46/50
2/2 [==============================] - 0s 198ms/step - loss: 1.5388 - accuracy: 0.7446 - val_loss:
1.5693 - val_accuracy: 0.7512
Epoch 47/50
2/2 [==============================] - 0s 197ms/step - loss: 1.5173 - accuracy: 0.7468 - val_loss:
1.5512 - val_accuracy: 0.7488
Epoch 48/50
2/2 [==============================] - 0s 197ms/step - loss: 1.4945 - accuracy: 0.7496 - val_loss:
1.5388 - val_accuracy: 0.7512
Epoch 49/50
2/2 [==============================] - 0s 203ms/step - loss: 1.4740 - accuracy: 0.7514 - val_loss:
1.5251 - val_accuracy: 0.7523
Epoch 50/50

```
Epoch 50/50
2/2 [==============================] - 0s 198ms/step - loss: 1.4522 - accuracy: 0.7538 - val_loss:
1.5122 - val_accuracy: 0.7535
```

Out[77]:

```
<tensorflow.python.keras.callbacks.History at 0x7fdd62ee5c90>
```

In [78]:

```
model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=50,
          validation_data=(source_padded_docs_test,target_padded_docs_test))
```

```
Epoch 1/50
2/2 [==============================] - 0s 219ms/step - loss: 1.4318 - accuracy: 0.7570 - val_loss:
1.5012 - val_accuracy: 0.7558
Epoch 2/50
2/2 [==============================] - 0s 195ms/step - loss: 1.4119 - accuracy: 0.7588 - val_loss:
1.4879 - val_accuracy: 0.7581
Epoch 3/50
2/2 [==============================] - 0s 195ms/step - loss: 1.3912 - accuracy: 0.7616 - val_loss:
1.4809 - val_accuracy: 0.7570
Epoch 4/50
2/2 [==============================] - 0s 197ms/step - loss: 1.3718 - accuracy: 0.7639 - val_loss:
1.4719 - val_accuracy: 0.7593
Epoch 5/50
2/2 [==============================] - 0s 197ms/step - loss: 1.3536 - accuracy: 0.7662 - val_loss:
1.4642 - val_accuracy: 0.7640
Epoch 6/50
2/2 [==============================] - 0s 197ms/step - loss: 1.3356 - accuracy: 0.7684 - val_loss:
1.4547 - val_accuracy: 0.7640
Epoch 7/50
2/2 [==============================] - 0s 195ms/step - loss: 1.3175 - accuracy: 0.7714 - val_loss:
1.4461 - val_accuracy: 0.7663
Epoch 8/50
2/2 [==============================] - 0s 194ms/step - loss: 1.3049 - accuracy: 0.7743 - val_loss:
1.4354 - val_accuracy: 0.7709
Epoch 9/50
2/2 [==============================] - 0s 194ms/step - loss: 1.2862 - accuracy: 0.7749 - val_loss:
1.4347 - val_accuracy: 0.7698
Epoch 10/50
2/2 [==============================] - 0s 195ms/step - loss: 1.2666 - accuracy: 0.7775 - val_loss:
1.4219 - val_accuracy: 0.7756
Epoch 11/50
2/2 [==============================] - 0s 198ms/step - loss: 1.2512 - accuracy: 0.7798 - val_loss:
1.4201 - val_accuracy: 0.7767
Epoch 12/50
2/2 [==============================] - 0s 197ms/step - loss: 1.2332 - accuracy: 0.7818 - val_loss:
1.4153 - val_accuracy: 0.7837
Epoch 13/50
2/2 [==============================] - 0s 197ms/step - loss: 1.2179 - accuracy: 0.7837 - val_loss:
1.4051 - val_accuracy: 0.7826
Epoch 14/50
2/2 [==============================] - 0s 197ms/step - loss: 1.2040 - accuracy: 0.7852 - val_loss:
1.4050 - val_accuracy: 0.7814
Epoch 15/50
2/2 [==============================] - 0s 194ms/step - loss: 1.1877 - accuracy: 0.7877 - val_loss:
1.4030 - val_accuracy: 0.7767
Epoch 16/50
2/2 [==============================] - 0s 196ms/step - loss: 1.1736 - accuracy: 0.7891 - val_loss:
1.3962 - val_accuracy: 0.7814
Epoch 17/50
2/2 [==============================] - 0s 200ms/step - loss: 1.1605 - accuracy: 0.7903 - val_loss:
1.3934 - val_accuracy: 0.7826
Epoch 18/50
2/2 [==============================] - 0s 195ms/step - loss: 1.1446 - accuracy: 0.7924 - val_loss:
1.3894 - val_accuracy: 0.7826
Epoch 19/50
2/2 [==============================] - 0s 195ms/step - loss: 1.1321 - accuracy: 0.7928 - val_loss:
1.3856 - val_accuracy: 0.7849
Epoch 20/50
2/2 [==============================] - 0s 197ms/step - loss: 1.1188 - accuracy: 0.7956 - val_loss:
1.3801 - val_accuracy: 0.7860
Epoch 21/50
2/2 [==============================] - 0s 195ms/step - loss: 1.1059 - accuracy: 0.7968 - val_loss:
1.3851 - val_accuracy: 0.7837
Epoch 22/50
2/2 [==============================] - 0s 195ms/step - loss: 1.0959 - accuracy: 0.7984 - val_loss:
1.3745 - val_accuracy: 0.7849
Epoch 23/50
2/2 [==============================] - 0s 197ms/step - loss: 1.0825 - accuracy: 0.7997 - val_loss:
1.3753 - val_accuracy: 0.7884
```

```
1.3755 - val_accuracy: 0.7901
Epoch 24/50
2/2 [==============================] - 0s 195ms/step - loss: 1.0683 - accuracy: 0.8015 - val_loss:
1.3692 - val_accuracy: 0.7872
Epoch 25/50
2/2 [==============================] - 0s 200ms/step - loss: 1.0570 - accuracy: 0.8034 - val_loss:
1.3686 - val_accuracy: 0.7860
Epoch 26/50
2/2 [==============================] - 0s 197ms/step - loss: 1.0461 - accuracy: 0.8049 - val_loss:
1.3703 - val_accuracy: 0.7837
Epoch 27/50
2/2 [==============================] - 0s 198ms/step - loss: 1.0450 - accuracy: 0.8061 - val_loss:
1.3621 - val_accuracy: 0.7802
Epoch 28/50
2/2 [==============================] - 0s 197ms/step - loss: 1.0258 - accuracy: 0.8076 - val_loss:
1.3689 - val_accuracy: 0.7919
Epoch 29/50
2/2 [==============================] - 0s 199ms/step - loss: 1.0154 - accuracy: 0.8103 - val_loss:
1.3576 - val_accuracy: 0.7860
Epoch 30/50
2/2 [==============================] - 0s 198ms/step - loss: 1.0025 - accuracy: 0.8114 - val_loss:
1.3580 - val_accuracy: 0.7872
Epoch 31/50
2/2 [==============================] - 0s 210ms/step - loss: 0.9906 - accuracy: 0.8136 - val_loss:
1.3545 - val_accuracy: 0.7907
Epoch 32/50
2/2 [==============================] - 0s 195ms/step - loss: 0.9804 - accuracy: 0.8150 - val_loss:
1.3499 - val_accuracy: 0.7860
Epoch 33/50
2/2 [==============================] - 0s 195ms/step - loss: 0.9708 - accuracy: 0.8167 - val_loss:
1.3459 - val_accuracy: 0.7860
Epoch 34/50
2/2 [==============================] - 0s 197ms/step - loss: 0.9604 - accuracy: 0.8180 - val_loss:
1.3474 - val_accuracy: 0.7884
Epoch 35/50
2/2 [==============================] - 0s 200ms/step - loss: 0.9490 - accuracy: 0.8195 - val_loss:
1.3437 - val_accuracy: 0.7895
Epoch 36/50
2/2 [==============================] - 0s 197ms/step - loss: 0.9391 - accuracy: 0.8212 - val_loss:
1.3473 - val_accuracy: 0.7895
Epoch 37/50
2/2 [==============================] - 0s 197ms/step - loss: 0.9304 - accuracy: 0.8224 - val_loss:
1.3405 - val_accuracy: 0.7919
Epoch 38/50
2/2 [==============================] - 0s 203ms/step - loss: 0.9220 - accuracy: 0.8235 - val_loss:
1.3635 - val_accuracy: 0.7953
Epoch 39/50
2/2 [==============================] - 0s 197ms/step - loss: 0.9493 - accuracy: 0.8205 - val_loss:
1.3883 - val_accuracy: 0.7965
Epoch 40/50
2/2 [==============================] - 0s 196ms/step - loss: 0.9714 - accuracy: 0.8195 - val_loss:
1.3331 - val_accuracy: 0.7895
Epoch 41/50
2/2 [==============================] - 0s 201ms/step - loss: 0.9365 - accuracy: 0.8226 - val_loss:
1.3686 - val_accuracy: 0.7907
Epoch 42/50
2/2 [==============================] - 0s 196ms/step - loss: 0.9257 - accuracy: 0.8251 - val_loss:
1.3489 - val_accuracy: 0.7895
Epoch 43/50
2/2 [==============================] - 0s 198ms/step - loss: 0.9110 - accuracy: 0.8265 - val_loss:
1.3295 - val_accuracy: 0.7953
Epoch 44/50
2/2 [==============================] - 0s 199ms/step - loss: 0.9032 - accuracy: 0.8268 - val_loss:
1.3496 - val_accuracy: 0.7907
Epoch 45/50
2/2 [==============================] - 0s 198ms/step - loss: 0.8888 - accuracy: 0.8298 - val_loss:
1.3387 - val_accuracy: 0.7907
Epoch 46/50
2/2 [==============================] - 0s 199ms/step - loss: 0.8783 - accuracy: 0.8308 - val_loss:
1.3274 - val_accuracy: 0.7942
Epoch 47/50
2/2 [==============================] - 0s 198ms/step - loss: 0.8699 - accuracy: 0.8318 - val_loss:
1.3279 - val_accuracy: 0.7907
Epoch 48/50
2/2 [==============================] - 0s 198ms/step - loss: 0.8574 - accuracy: 0.8326 - val_loss:
1.3367 - val_accuracy: 0.7895
Epoch 49/50
2/2 [==============================] - 0s 195ms/step - loss: 0.8501 - accuracy: 0.8332 - val_loss:
```

```
2/2 [==============================] - 0s 195ms/step - loss: 0.8501 - accuracy: 0.8332 - val_loss:
1.3306 - val_accuracy: 0.7919
Epoch 50/50
2/2 [==============================] - 0s 196ms/step - loss: 0.8399 - accuracy: 0.8357 - val_loss:
1.3322 - val_accuracy: 0.7965
```

Out[78]:

```
<tensorflow.python.keras.callbacks.History at 0x7fdd62b6e990>
```

In [79]:

```
x=model.predict(source_padded_docs_test[:1])[0]
```

In [80]:

```
#https://machinelearningmastery.com

index_to_words = {id: word for word, id in target_tokenizer.word_index.items()}
index_to_words[0] = '<PAD>'

' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
```

Out[80]:

```
'hey i am still having breakfast <PAD> if you reach there first can help love and me want 12 <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PA
D> <PAD> <PAD> <PAD> <PAD>'
```

In [81]:

```
print(y_test[:1])
```

```
1866    I am still having breakfast. If you reach ther...
Name: target, dtype: object
```

In [82]:

```
X_test[:1]
```

Out[82]:

```
1866    Hey i am still having breakfast eh. If you rea...
Name: source, dtype: object
```

In [83]:

```
def prediction(x):

  index_to_words = {id: word for word, id in target_tokenizer.word_index.items()}
  index_to_words[0] = '<PAD>'

  y=' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
  return y
for i in range(20):
  print("Input text: ")
  a=list(X_test[i:i+1])
  print(a[0])

  print("Actual Output: ")
  b=list(y_test[i:i+1])
  print(b[0])

  print("Predicted Output: ")
  x=model.predict(source_padded_docs_test[i:i+1])
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  print(' '.join(y_lst))
  print('>'*180)
```

```
Input text:
Hey i am still having breakfast eh. If you reach there first can help rebecca and me chope seats?
Actual Output:
I am still having breakfast. If you reach there first can you help me and Rebecca reserve seats?
Predicted Output:
hey i am still having breakfast if you reach there first can help love and me want 12
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Huh ü take then how i take bus later... Inside got money a not...
Actual Output:
If you take then how I take bus later? Inside got money or not?
Predicted Output:
huh you take then how i take bus later then out have a not
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hi neva worry bout da truth coz the truth will lead me 2 ur heart. It's the least a unique person like u
deserve. Sleep tight or morning
Actual Output:
Hi, never worry about the truth because the truth will lead me to your heart. It's the least that a uniqu
e person like you deserve. Sleep tight or morning.
Predicted Output:
hi never worry about the because the will me for your your it's the is a exciting person like you you sl
eep have one another
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Take so long
Actual Output:
Take so long.
Predicted Output:
take so long
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey where r ü im here liao
Actual Output:
Hey, where are you? I'm here.
Predicted Output:
hey where are you i'm here
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hi everyone hows ur day ?
Actual Output:
Hi everyone, how's your day?
Predicted Output:
hi everyone how's your day
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha- if no need make up ñ near my wkplace ñ not wk too late.can consider.tt is if ü can find such a pla
ce.ay,abt a mth ago she say she wk ere la. Hee-
Actual Output:
Haha. If no need to make up and near my workplace and does not work too late. Can consider. That is if y
ou can find such a place. AY, about a month ago, she said she worked there.
Predicted Output:
haha if no need make up and near my and not week too late free you that is if you can find a this this g
ood a a a very very she very sleep but hehe
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
How i noe... Last time tis one is on offer wat...
Actual Output:
How I know. Last time this one is on offer.
Predicted Output:
how i know the time this time is is only what
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I reached already
Actual Output:
I reached already.
Predicted Output:
i reached already
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haiyoh... It was so crowded... We didnt buy anything... Haha... Lots of pple in town. So mon we go
facial with ü then go shopping?
Actual Output:
Ouch. It was so crowded. We didn't buy anything. Haha. There are lots of people in town. So Monday we go
facial with you then go shopping?
Predicted Output:
violyn it was so crowded we didn't buy anything haha good of people in town so monday we go with you
then go go
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
HI MERINA NICE 2 CHAT WITH U. UR HP NO PLS. WHAT IS UR RACE?
```

```
Actual Output:
Hi Merina. It's nice to chat with you. Your hand phone number please. What is your race?
Predicted Output:
hi dom nice to chat with you your your number please what is your your
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hmmm.... After my drivin den free lor... Y?
Actual Output:
After my driving then I will be free. Why?
Predicted Output:
hmm after my driving then free why
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Erm anything lor...Can bring tmr? Thx =)
Actual Output:
Can anything be brought tomorrow? Thanks.
Predicted Output:
no anything i can bring tomorrow thanks
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Okay... they arent open on public holidays
Actual Output:
Okay. They aren't open on public holidays.
Predicted Output:
ok they aren't open on
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha... I'm carrying a broom with me so really paiseh to walk into lecture with it. I'm coming straight
from home mah... Cya later then.
Actual Output:
Haha. I'm carrying a broom with me. So I'm really sorry to walk into lecture with it. I'm coming
straight from home. See you later then.
Predicted Output:
haha i'm a a me so really embarrassing to to into lecture with it off from home later later
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hmmm.... I'm watchin w my frens oredi... Paiseh...
Actual Output:
Hmm. I'm watching with my friends already. It's embarrassing.
Predicted Output:
hmm i'm watching with my friends already haha
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey... Ü 've got driving today? my driving at 240.
Actual Output:
Hey. You have got driving today? My driving is at 2:40.
Predicted Output:
hey you got driving today my driving at 15 17
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
then it can moisturise our skin. and rub in circular motion. u wash face, tone,then put a bit of jelly a
nd cream onto ur hand,and tap it on your face,
Actual Output:
Then it can moisturise our skin and rub in circular motion. You wash face, tone, then put a bit of jelly
and cream onto your hand, and tap it on your face.
Predicted Output:
then it can our skin and in you and and then then a bit of and and your and it on your eyes
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I'm pubbin now, gee, cant go online...After my drivin ah, hmmm, den where ur meetin....
Actual Output:
I'm in pub now. I can't go online. After my driving, then where are you meeting?
Predicted Output:
violyn i'm now why can't go go after my driving i i then where you are
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha... Not accurate right....
Actual Output:
```

```
Haha. Not accurate, right?
Predicted Output:
haha not right
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [84]:

```
import nltk.translate.bleu_score as bleu

bleu_score=[]
for i in range(20):
  b=list(y_test[i:i+1])
  x=model.predict(source_padded_docs_test[i:i+1])
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  bleu_score.append(bleu.sentence_bleu([b[0].split(),],y_lst))
print(bleu_score)
print("The Average Bleu Score is: ",sum(bleu_score)/20)
```

```
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
[0.2911892337378932, 0.21142141714303078, 0.14463972129203845, 0.7598356856515925, 0.5081327481546147,
0.5623413251903491, 0.21662732770853732, 0.4063798282013443, 0.7598356856515925, 0.16504659724801518,
0.17795291340072017, 0.30895757752065417, 0.6147881529512643, 0.3769486629893372, 0.1584557519176515,
0.3050975216056289, 0.41545589177443254, 0.28513533990048395, 0.4914498405430853, 0]
The Average Bleu Score is:  0.35798456112911325
```

Model2:

In [103]:

```
input=tf.keras.layers.Input(shape=(43,))
embed=tf.keras.layers.Embedding(source_vocab_size,300,weights=[embedding_matrix],input_length=source_padc
lstm1=tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128, return_sequences=True))(embed)
output=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(target_vocab_size, activation='softmax'))(ls
model=tf.keras.models.Model(inputs=input,outputs=output)
model.summary()
```

```
Model: "model_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 43)]              0
_________________________________________________________________
embedding_12 (Embedding)     (None, 43, 300)           1111200
_________________________________________________________________
bidirectional_8 (Bidirection (None, 43, 256)           439296
_________________________________________________________________
time_distributed_17 (TimeDis (None, 43, 3033)          779481
=================================================================
Total params: 2,329,977
Trainable params: 1,218,777
Non-trainable params: 1,111,200
_________________________________________________________________
```

In [104]:

```
# Compile model
model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
              loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

In [105]:

```
model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=50,
         validation_data=(source_padded_docs_test,target_padded_docs_test))
```

```
Epoch 1/50
```

```
2/2 [==============================] - 3s 849ms/step - loss: 7.9556 - accuracy: 0.3262 - val_loss:
6.6151 - val_accuracy: 0.6907
Epoch 2/50
2/2 [==============================] - 0s 241ms/step - loss: 5.7774 - accuracy: 0.6827 - val_loss:
3.0524 - val_accuracy: 0.6814
Epoch 3/50
2/2 [==============================] - 0s 255ms/step - loss: 3.1042 - accuracy: 0.6737 - val_loss:
3.1649 - val_accuracy: 0.6814
Epoch 4/50
2/2 [==============================] - 0s 243ms/step - loss: 3.2320 - accuracy: 0.6737 - val_loss:
2.7467 - val_accuracy: 0.6814
Epoch 5/50
2/2 [==============================] - 0s 246ms/step - loss: 2.7685 - accuracy: 0.6747 - val_loss:
2.4998 - val_accuracy: 0.6837
Epoch 6/50
2/2 [==============================] - 0s 240ms/step - loss: 2.6061 - accuracy: 0.6786 - val_loss:
2.3973 - val_accuracy: 0.6895
Epoch 7/50
2/2 [==============================] - 0s 248ms/step - loss: 2.4415 - accuracy: 0.6799 - val_loss:
2.2410 - val_accuracy: 0.6895
Epoch 8/50
2/2 [==============================] - 0s 247ms/step - loss: 2.3152 - accuracy: 0.6780 - val_loss:
2.1987 - val_accuracy: 0.6907
Epoch 9/50
2/2 [==============================] - 0s 245ms/step - loss: 2.2456 - accuracy: 0.6825 - val_loss:
2.0994 - val_accuracy: 0.6953
Epoch 10/50
2/2 [==============================] - 0s 248ms/step - loss: 2.1551 - accuracy: 0.6886 - val_loss:
2.0266 - val_accuracy: 0.6942
Epoch 11/50
2/2 [==============================] - 0s 243ms/step - loss: 2.0836 - accuracy: 0.6907 - val_loss:
1.9746 - val_accuracy: 0.6953
Epoch 12/50
2/2 [==============================] - 0s 250ms/step - loss: 2.0330 - accuracy: 0.6924 - val_loss:
1.9445 - val_accuracy: 0.6953
Epoch 13/50
2/2 [==============================] - 0s 247ms/step - loss: 2.0021 - accuracy: 0.6946 - val_loss:
1.9227 - val_accuracy: 0.6988
Epoch 14/50
2/2 [==============================] - 0s 251ms/step - loss: 1.9777 - accuracy: 0.6968 - val_loss:
1.8978 - val_accuracy: 0.6965
Epoch 15/50
2/2 [==============================] - 0s 254ms/step - loss: 1.9512 - accuracy: 0.6977 - val_loss:
1.8775 - val_accuracy: 0.6965
Epoch 16/50
2/2 [==============================] - 0s 248ms/step - loss: 1.9257 - accuracy: 0.6984 - val_loss:
1.8562 - val_accuracy: 0.6977
Epoch 17/50
2/2 [==============================] - 0s 251ms/step - loss: 1.8984 - accuracy: 0.6995 - val_loss:
1.8364 - val_accuracy: 0.7012
Epoch 18/50
2/2 [==============================] - 0s 252ms/step - loss: 1.8719 - accuracy: 0.7014 - val_loss:
1.8179 - val_accuracy: 0.7058
Epoch 19/50
2/2 [==============================] - 0s 255ms/step - loss: 1.8448 - accuracy: 0.7038 - val_loss:
1.7973 - val_accuracy: 0.7116
Epoch 20/50
2/2 [==============================] - 0s 249ms/step - loss: 1.8165 - accuracy: 0.7067 - val_loss:
1.7767 - val_accuracy: 0.7140
Epoch 21/50
2/2 [==============================] - 0s 252ms/step - loss: 1.7868 - accuracy: 0.7098 - val_loss:
1.7547 - val_accuracy: 0.7163
Epoch 22/50
2/2 [==============================] - 0s 246ms/step - loss: 1.7564 - accuracy: 0.7134 - val_loss:
1.7322 - val_accuracy: 0.7233
Epoch 23/50
2/2 [==============================] - 0s 250ms/step - loss: 1.7239 - accuracy: 0.7176 - val_loss:
1.7102 - val_accuracy: 0.7291
Epoch 24/50
2/2 [==============================] - 0s 248ms/step - loss: 1.6907 - accuracy: 0.7226 - val_loss:
1.6875 - val_accuracy: 0.7360
Epoch 25/50
2/2 [==============================] - 0s 247ms/step - loss: 1.6557 - accuracy: 0.7277 - val_loss:
1.6653 - val_accuracy: 0.7395
Epoch 26/50
2/2 [==============================] - 0s 251ms/step - loss: 1.6198 - accuracy: 0.7327 - val_loss:
1.6416 - val accuracy: 0.7442
```

```
Epoch 27/50
2/2 [==============================] - 0s 252ms/step - loss: 1.5834 - accuracy: 0.7383 - val_loss:
1.6168 - val_accuracy: 0.7419
Epoch 28/50
2/2 [==============================] - 0s 251ms/step - loss: 1.5469 - accuracy: 0.7431 - val_loss:
1.5945 - val_accuracy: 0.7453
Epoch 29/50
2/2 [==============================] - 0s 251ms/step - loss: 1.5101 - accuracy: 0.7476 - val_loss:
1.5752 - val_accuracy: 0.7535
Epoch 30/50
2/2 [==============================] - 0s 249ms/step - loss: 1.4734 - accuracy: 0.7526 - val_loss:
1.5530 - val_accuracy: 0.7570
Epoch 31/50
2/2 [==============================] - 0s 252ms/step - loss: 1.4366 - accuracy: 0.7575 - val_loss:
1.5287 - val_accuracy: 0.7605
Epoch 32/50
2/2 [==============================] - 0s 251ms/step - loss: 1.4002 - accuracy: 0.7624 - val_loss:
1.5085 - val_accuracy: 0.7651
Epoch 33/50
2/2 [==============================] - 0s 249ms/step - loss: 1.3639 - accuracy: 0.7669 - val_loss:
1.4891 - val_accuracy: 0.7698
Epoch 34/50
2/2 [==============================] - 0s 253ms/step - loss: 1.3284 - accuracy: 0.7717 - val_loss:
1.4709 - val_accuracy: 0.7744
Epoch 35/50
2/2 [==============================] - 1s 254ms/step - loss: 1.2938 - accuracy: 0.7771 - val_loss:
1.4579 - val_accuracy: 0.7744
Epoch 36/50
2/2 [==============================] - 0s 250ms/step - loss: 1.2605 - accuracy: 0.7818 - val_loss:
1.4413 - val_accuracy: 0.7802
Epoch 37/50
2/2 [==============================] - 0s 254ms/step - loss: 1.2287 - accuracy: 0.7859 - val_loss:
1.4324 - val_accuracy: 0.7837
Epoch 38/50
2/2 [==============================] - 0s 252ms/step - loss: 1.1971 - accuracy: 0.7907 - val_loss:
1.4179 - val_accuracy: 0.7849
Epoch 39/50
2/2 [==============================] - 0s 252ms/step - loss: 1.1672 - accuracy: 0.7946 - val_loss:
1.4094 - val_accuracy: 0.7872
Epoch 40/50
2/2 [==============================] - 0s 253ms/step - loss: 1.1382 - accuracy: 0.7996 - val_loss:
1.3959 - val_accuracy: 0.7884
Epoch 41/50
2/2 [==============================] - 0s 247ms/step - loss: 1.1106 - accuracy: 0.8030 - val_loss:
1.3886 - val_accuracy: 0.7907
Epoch 42/50
2/2 [==============================] - 1s 256ms/step - loss: 1.0839 - accuracy: 0.8079 - val_loss:
1.3777 - val_accuracy: 0.7872
Epoch 43/50
2/2 [==============================] - 1s 251ms/step - loss: 1.0590 - accuracy: 0.8113 - val_loss:
1.3712 - val_accuracy: 0.7860
Epoch 44/50
2/2 [==============================] - 1s 252ms/step - loss: 1.0351 - accuracy: 0.8151 - val_loss:
1.3662 - val_accuracy: 0.7872
Epoch 45/50
2/2 [==============================] - 0s 254ms/step - loss: 1.0116 - accuracy: 0.8187 - val_loss:
1.3628 - val_accuracy: 0.7872
Epoch 46/50
2/2 [==============================] - 0s 256ms/step - loss: 0.9898 - accuracy: 0.8212 - val_loss:
1.3549 - val_accuracy: 0.7884
Epoch 47/50
2/2 [==============================] - 0s 256ms/step - loss: 0.9694 - accuracy: 0.8238 - val_loss:
1.3513 - val_accuracy: 0.7884
Epoch 48/50
2/2 [==============================] - 1s 255ms/step - loss: 0.9498 - accuracy: 0.8262 - val_loss:
1.3483 - val_accuracy: 0.7919
Epoch 49/50
2/2 [==============================] - 0s 253ms/step - loss: 0.9304 - accuracy: 0.8280 - val_loss:
1.3375 - val_accuracy: 0.7919
Epoch 50/50
2/2 [==============================] - 1s 263ms/step - loss: 0.9133 - accuracy: 0.8294 - val_loss:
1.3405 - val_accuracy: 0.7942
```

Out[105]:

```
<tensorflow.python.keras.callbacks.History at 0x7fdd575f1dd0>
```

In [106]:

```
x=model.predict(source_padded_docs_test[7:8])[0]
```

In [107]:

```
index_to_words = {id: word for word, id in target_tokenizer.word_index.items()}
index_to_words[0] = '<PAD>'

' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
```

Out[107]:

```
'how i know i time this one is on <PAD> what <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD>
<PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PAD> <PA
D> <PAD> <PAD> <PAD> <PAD>'
```

In [108]:

```
print(y_test[7:8])
```

```
128    How I know. Last time this one is on offer.
Name: target, dtype: object
```

In [109]:

```
X_test[7:8]
```

Out[109]:

```
128    How i noe... Last time tis one is on offer wat...
Name: source, dtype: object
```

In [110]:

```
def prediction(x):

  index_to_words = {id: word for word, id in target_tokenizer.word_index.items()}
  index_to_words[0] = '<PAD>'

  y=' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
  return y
for i in range(20):
  print("Input text: ")
  a=list(X_test[i:i+1])
  print(a[0])

  print("Actual Output: ")
  b=list(y_test[i:i+1])
  print(b[0])

  print("Predicted Output: ")
  x=model.predict(source_padded_docs_test[i:i+1])
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  print(' '.join(y_lst))
  print('>'*180)
```

```
Input text:
Hey i am still having breakfast eh. If you reach there first can help rebecca and me chope seats?
Actual Output:
I am still having breakfast. If you reach there first can you help me and Rebecca reserve seats?
Predicted Output:
hey i am still having day i if you reach there the can help michelle and me buy
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Huh ü take then how i take bus later... Inside got money a not...
Actual Output:
If you take then how I take bus later? Inside got money or not?
Predicted Output:
huh you you then how i take bus later and got money a not
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hi neva worry bout da truth coz the truth will lead me 2 ur heart. It's the least a unique person like u
deserve. Sleep tight or morning
Actual Output:
Hi, never worry about the truth because the truth will lead me to your heart. It's the least that a uniqu
e person like you deserve. Sleep tight or morning.
Predicted Output:
hi never worry about the because the will me for your heart it's the one a a person like you you sleep g
ood or morning
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Take so long
Actual Output:
Take so long.
Predicted Output:
take so long
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey where r ü im here liao
Actual Output:
Hey, where are you? I'm here.
Predicted Output:
hey where are you i'm here
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hi everyone hows ur day ?
Actual Output:
Hi everyone, how's your day?
Predicted Output:
hi everyone how's your day
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha- if no need make up ñ near my wkplace ñ not wk too late.can consider.tt is if ü can find such a pla
ce.ay,abt a mth ago she say she wk ere la. Hee-
Actual Output:
Haha. If no need to make up and near my workplace and does not work too late. Can consider. That is if y
ou can find such a place. AY, about a month ago, she said she worked there.
Predicted Output:
haha if no need make up and near my and not month so late can i that is if you can find a place you abou
t a month ago she says she good not hee
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
How i noe... Last time tis one is on offer wat...
Actual Output:
How I know. Last time this one is on offer.
Predicted Output:
how i know i time this one is on what
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I reached already
Actual Output:
I reached already.
Predicted Output:
i reached already
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haiyoh... It was so crowded... We didnt buy anything... Haha... Lots of pple in town. So mon we go
facial with ü then go shopping?
Actual Output:
Ouch. It was so crowded. We didn't buy anything. Haha. There are lots of people in town. So Monday we go
facial with you then go shopping?
Predicted Output:
yes it was so crowded we didn't buy anything haha of of people in town so so we go and you then go
shopping
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
HI MERINA NICE 2 CHAT WITH U. UR HP NO PLS. WHAT IS UR RACE?
Actual Output:
Hi Merina. It's nice to chat with you. Your hand phone number please. What is your race?
Predicted Output:
hi merina nice to chat with you your hand number please what is your
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hmmm.... After my drivin den free lor... Y?
Actual Output:
After my driving then I will be free. Why?
Predicted Output:
```

```
hmm after my driving then free why
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Erm anything lor...Can bring tmr? Thx =)
Actual Output:
Can anything be brought tomorrow? Thanks.
Predicted Output:
er anything you can bring tomorrow thanks
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Okay... they arent open on public holidays
Actual Output:
Okay. They aren't open on public holidays.
Predicted Output:
ok they aren't open on
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha... I'm carrying a broom with me so really paiseh to walk into lecture with it. I'm coming straight
from home mah... Cya later then.
Actual Output:
Haha. I'm carrying a broom with me. So I'm really sorry to walk into lecture with it. I'm coming
straight from home. See you later then.
Predicted Output:
haha i'm a with me so really to to walk in lecture for this coming on from home my tomorrow later then
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hmmm.... I'm watchin w my frens oredi... Paiseh...
Actual Output:
Hmm. I'm watching with my friends already. It's embarrassing.
Predicted Output:
hmm i'm watching with my friend already embarrassing
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey... Ü 've got driving today? my driving at 240.
Actual Output:
Hey. You have got driving today? My driving is at 2:40.
Predicted Output:
hey you got driving today my driving at total
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
then it can moisturise our skin. and rub in circular motion. u wash face, tone,then put a bit of jelly a
nd cream onto ur hand,and tap it on your face,
Actual Output:
Then it can moisturise our skin and rub in circular motion. You wash face, tone, then put a bit of jelly
and cream onto your hand, and tap it on your face.
Predicted Output:
then it can our skin and and you to and then the a a of the the your the and the on your
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I'm pubbin now, gee, cant go online...After my drivin ah, hmmm, den where ur meetin....
Actual Output:
I'm in pub now. I can't go online. After my driving, then where are you meeting?
Predicted Output:
yes yes now i can't go in after my driving my i then where your to
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha... Not accurate right....
Actual Output:
Haha. Not accurate, right?
Predicted Output:
haha not right
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [111]:

```
import nltk.translate.bleu_score as bleu

bleu_score=[]
```

```
for i in range(20):
  b=list(y_test[i:i+1])
  x=model.predict(source_padded_docs_test[i:i+1])
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  bleu_score.append(bleu.sentence_bleu([b[0].split(),],y_lst))
print(bleu_score)
print("The Average Bleu Score is: ",sum(bleu_score)/20)
```

```
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 4-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:490: UserWarning:
Corpus/Sentence contains 0 counts of 2-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
[0.3575297164449809, 0.5828233954152654, 0.14715613818513176, 0.7598356856515925, 0.5081327481546147,
0.5623413251903491, 0.2428662778132657, 0.392814650900513, 0.7598356856515925, 0.3026565453571514,
0.18336673852940621, 0.30895757752065417, 0.6147881529512643, 0.3769486629893372, 0.3114852603245108,
0.32260135189272865, 0.38875142041440197, 0.1788409894677479, 0.4728708045015879, 0]
The Average Bleu Score is:  0.3887301563678047
```

In [ ]: