

In [1]:

```
!gdown --id 1OurDQutbWQacvT32HMqFL7vIUrSM1lOp
Downloading...
From: https://drive.google.com/uc?id=1OurDQutbWQacvT32HMqFL7vIUrSM1lOp
To: /content/preprocessed_data.csv
100% 300k/300k [00:00<00:00, 43.5MB/s]
```

In [2]:

```
#Importing the necessary packages
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 the file preprocessed_data.csv
```

In [4]:

```
df.head(4)#visulazing the DataFrame
```

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):#for removingt the last character
    x=x[:-1]
    return x
```

In [6]:

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

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
```

Out[8]:

```
(2000, 2)
```

In [9]:

```
df=df[df['source'].apply(len)<170]#removing sentences where source sentence is greater than 170
df=df[df['target'].apply(len)<200]#removing snetences where target sentence is greater than 200
```

In [10]:

```
df.shape#printing the shape
```

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 in the data in the ratio of
print(X_train.shape)
```

```
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Target:

In [12]:

```
target_tokenizer=Tokenizer(filters=None,char_level=True,lower=False)#tokenzing the target in character l
target_tokenizer.fit_on_texts(y_train)#fitting on the target train
target_vocab_size= len(target_tokenizer.word_index) + 1
print(len(target_tokenizer.word_index))#printing the vocabulary size
```

90

In [13]:

```
target_encoded_docs_train = target_tokenizer.texts_to_sequences(y_train)#converting target train into seq
target_encoded_docs_test = target_tokenizer.texts_to_sequences(y_test)#converting target test into sequer

target_padded_docs_train = pad_sequences(target_encoded_docs_train,padding='post')#padding target train
target_padded_docs_test = pad_sequences(target_encoded_docs_test,maxlen=target_padded_docs_train.shape[1])
```

Source:

In [14]:

```
source_tokenizer=Tokenizer(char_level=True,lower=False)#tokenzing the source in character level
source_tokenizer.fit_on_texts(X_train)#fitting on the source train
source_vocab_size= len(source_tokenizer.word_index) + 1
print(len(source_tokenizer.word_index))#printing the vocabulary size
```

103

In [15]:

```
source_encoded_docs_train = source_tokenizer.texts_to_sequences(X_train)#converting source train into seq
source_encoded_docs_test = source_tokenizer.texts_to_sequences(X_test)#converting source train into seque

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

In [16]:

```
#we are reshaping because sparse_categorical_entropy expects 3dimensions
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 [17]:

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

(1970, 199, 1)
(20, 199, 1)
```

In [18]:

```
#we are reshaping because sparse_categorical_entropy expects 3dimensions
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 [19]:

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

(1970, 199, 1)
(20, 199, 1)
```

In [20]:

```
X_train.to_csv('X_train2.csv')
y_train.to_csv('y_train2.csv')
X_test.to_csv('X_test2.csv')
y_test.to_csv('y_test2.csv')
```

In [21]:

```
import pandas as pd
pd.DataFrame(source_encoded_docs_train).to_csv("source_encoded_docs_train2.csv")
pd.DataFrame(source_encoded_docs_test).to_csv("source_encoded_docs_test2.csv")
pd.DataFrame(target_encoded_docs_train).to_csv("target_encoded_docs_train2.csv")
pd.DataFrame(target_encoded_docs_test).to_csv("target_encoded_docs_test2.csv")
```

Model1:

In [22]:

```
input=tf.keras.layers.Input(shape=(199,))
embed=tf.keras.layers.Embedding(source_vocab_size,256, input_length=source_padded_docs_train.shape[1])(in
```

```

lstm1=tf.keras.layers.LSTM(128, return_sequences=True)(embed)
dense=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(512, activation='relu'))(lstm1)
drop=tf.keras.layers.Dropout(0.5)(dense)
output=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(target_vocab_size, activation='softmax'))(dr
model=tf.keras.models.Model(inputs=input,outputs=output)
model.summary()

```

Model: "model"

| Layer (type)                 | Output Shape     | Param # |
|------------------------------|------------------|---------|
| input_1 (InputLayer)         | [(None, 199)]    | 0       |
| embedding (Embedding)        | (None, 199, 256) | 26624   |
| lstm (LSTM)                  | (None, 199, 128) | 197120  |
| time_distributed (TimeDistri | (None, 199, 512) | 66048   |
| dropout (Dropout)            | (None, 199, 512) | 0       |
| time_distributed_1 (TimeDist | (None, 199, 91)  | 46683   |
| Total params: 336,475        |                  |         |
| Trainable params: 336,475    |                  |         |
| Non-trainable params: 0      |                  |         |

In [23]:

```

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

```

In [24]:

```

model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=100,
          validation_data=(source_padded_docs_test,target_padded_docs_test))

```

```

Epoch 1/100
2/2 [=====] - 9s 1s/step - loss: 4.1856 - accuracy: 0.3459 - val_loss: 2.7667 -
val_accuracy: 0.6420
Epoch 2/100
2/2 [=====] - 1s 520ms/step - loss: 2.1551 - accuracy: 0.6701 - val_loss:
1.4970 - val_accuracy: 0.6907
Epoch 3/100
2/2 [=====] - 1s 532ms/step - loss: 1.4772 - accuracy: 0.6788 - val_loss:
1.3300 - val_accuracy: 0.6593
Epoch 4/100
2/2 [=====] - 1s 545ms/step - loss: 1.3352 - accuracy: 0.6729 - val_loss:
1.3038 - val_accuracy: 0.6965
Epoch 5/100
2/2 [=====] - 1s 501ms/step - loss: 1.3312 - accuracy: 0.6997 - val_loss:
1.3129 - val_accuracy: 0.6965
Epoch 6/100
2/2 [=====] - 1s 518ms/step - loss: 1.3253 - accuracy: 0.6968 - val_loss:
1.3014 - val_accuracy: 0.6965
Epoch 7/100
2/2 [=====] - 1s 515ms/step - loss: 1.3145 - accuracy: 0.6953 - val_loss:
1.2833 - val_accuracy: 0.6965
Epoch 8/100
2/2 [=====] - 1s 517ms/step - loss: 1.3080 - accuracy: 0.7002 - val_loss:
1.2783 - val_accuracy: 0.6965
Epoch 9/100
2/2 [=====] - 1s 520ms/step - loss: 1.2955 - accuracy: 0.7001 - val_loss:
1.2808 - val_accuracy: 0.6965
Epoch 10/100
2/2 [=====] - 1s 529ms/step - loss: 1.2924 - accuracy: 0.7000 - val_loss:
1.2741 - val_accuracy: 0.6965
Epoch 11/100
2/2 [=====] - 1s 534ms/step - loss: 1.2838 - accuracy: 0.6999 - val_loss:
1.2633 - val_accuracy: 0.6950
Epoch 12/100
2/2 [=====] - 1s 529ms/step - loss: 1.2738 - accuracy: 0.6996 - val_loss:
1.2536 - val_accuracy: 0.6950
Epoch 13/100
2/2 [=====] - 1s 539ms/step - loss: 1.2589 - accuracy: 0.6995 - val_loss:
1.2441 - val_accuracy: 0.6962
Epoch 14/100
2/2 [=====] - 1s 522ms/step - loss: 1.2412 - accuracy: 0.6999 - val_loss:
1.2363 - val accuracy: 0.6960

```

Epoch 15/100  
2/2 [=====] - 1s 531ms/step - loss: 1.2297 - accuracy: 0.7002 - val\_loss: 1.2348 - val\_accuracy: 0.6965  
Epoch 16/100  
2/2 [=====] - 1s 533ms/step - loss: 1.2226 - accuracy: 0.7008 - val\_loss: 1.2262 - val\_accuracy: 0.6970  
Epoch 17/100  
2/2 [=====] - 1s 530ms/step - loss: 1.2173 - accuracy: 0.7013 - val\_loss: 1.2253 - val\_accuracy: 0.6987  
Epoch 18/100  
2/2 [=====] - 1s 524ms/step - loss: 1.2148 - accuracy: 0.7018 - val\_loss: 1.2243 - val\_accuracy: 0.6992  
Epoch 19/100  
2/2 [=====] - 1s 533ms/step - loss: 1.2106 - accuracy: 0.7019 - val\_loss: 1.2291 - val\_accuracy: 0.6992  
Epoch 20/100  
2/2 [=====] - 1s 535ms/step - loss: 1.2080 - accuracy: 0.7019 - val\_loss: 1.2227 - val\_accuracy: 0.6992  
Epoch 21/100  
2/2 [=====] - 1s 525ms/step - loss: 1.2045 - accuracy: 0.7020 - val\_loss: 1.2185 - val\_accuracy: 0.6990  
Epoch 22/100  
2/2 [=====] - 1s 518ms/step - loss: 1.2008 - accuracy: 0.7021 - val\_loss: 1.2257 - val\_accuracy: 0.6992  
Epoch 23/100  
2/2 [=====] - 1s 533ms/step - loss: 1.2082 - accuracy: 0.7020 - val\_loss: 1.2150 - val\_accuracy: 0.6990  
Epoch 24/100  
2/2 [=====] - 1s 498ms/step - loss: 1.2034 - accuracy: 0.7023 - val\_loss: 1.2149 - val\_accuracy: 0.6995  
Epoch 25/100  
2/2 [=====] - 1s 497ms/step - loss: 1.2005 - accuracy: 0.7024 - val\_loss: 1.2112 - val\_accuracy: 0.6992  
Epoch 26/100  
2/2 [=====] - 1s 510ms/step - loss: 1.1942 - accuracy: 0.7027 - val\_loss: 1.2096 - val\_accuracy: 0.7003  
Epoch 27/100  
2/2 [=====] - 1s 526ms/step - loss: 1.1917 - accuracy: 0.7029 - val\_loss: 1.2017 - val\_accuracy: 0.7005  
Epoch 28/100  
2/2 [=====] - 1s 548ms/step - loss: 1.1871 - accuracy: 0.7032 - val\_loss: 1.2008 - val\_accuracy: 0.7000  
Epoch 29/100  
2/2 [=====] - 1s 520ms/step - loss: 1.1816 - accuracy: 0.7032 - val\_loss: 1.1972 - val\_accuracy: 0.7003  
Epoch 30/100  
2/2 [=====] - 1s 509ms/step - loss: 1.1772 - accuracy: 0.7034 - val\_loss: 1.2440 - val\_accuracy: 0.7013  
Epoch 31/100  
2/2 [=====] - 1s 521ms/step - loss: 1.2147 - accuracy: 0.7034 - val\_loss: 1.2044 - val\_accuracy: 0.7005  
Epoch 32/100  
2/2 [=====] - 1s 533ms/step - loss: 1.1946 - accuracy: 0.7036 - val\_loss: 1.2172 - val\_accuracy: 0.7003  
Epoch 33/100  
2/2 [=====] - 1s 522ms/step - loss: 1.1996 - accuracy: 0.7040 - val\_loss: 1.2092 - val\_accuracy: 0.7018  
Epoch 34/100  
2/2 [=====] - 1s 537ms/step - loss: 1.1859 - accuracy: 0.7043 - val\_loss: 1.2110 - val\_accuracy: 0.7003  
Epoch 35/100  
2/2 [=====] - 1s 527ms/step - loss: 1.1862 - accuracy: 0.7041 - val\_loss: 1.1951 - val\_accuracy: 0.7015  
Epoch 36/100  
2/2 [=====] - 1s 532ms/step - loss: 1.1761 - accuracy: 0.7048 - val\_loss: 1.2010 - val\_accuracy: 0.7028  
Epoch 37/100  
2/2 [=====] - 1s 524ms/step - loss: 1.1781 - accuracy: 0.7052 - val\_loss: 1.1944 - val\_accuracy: 0.7043  
Epoch 38/100  
2/2 [=====] - 1s 525ms/step - loss: 1.1702 - accuracy: 0.7062 - val\_loss: 1.1933 - val\_accuracy: 0.7033  
Epoch 39/100  
2/2 [=====] - 1s 528ms/step - loss: 1.1704 - accuracy: 0.7062 - val\_loss: 1.1935 - val\_accuracy: 0.7050  
Epoch 40/100  
2/2 [=====] - 1s 541ms/step - loss: 1.1640 - accuracy: 0.7071 - val\_loss:

1.1910 - val\_accuracy: 0.7050  
Epoch 41/100  
2/2 [=====] - 1s 510ms/step - loss: 1.1611 - accuracy: 0.7075 - val\_loss:  
1.1822 - val\_accuracy: 0.7048  
Epoch 42/100  
2/2 [=====] - 1s 508ms/step - loss: 1.1574 - accuracy: 0.7079 - val\_loss:  
1.1742 - val\_accuracy: 0.7053  
Epoch 43/100  
2/2 [=====] - 1s 538ms/step - loss: 1.1548 - accuracy: 0.7080 - val\_loss:  
1.1724 - val\_accuracy: 0.7048  
Epoch 44/100  
2/2 [=====] - 1s 531ms/step - loss: 1.1513 - accuracy: 0.7088 - val\_loss:  
1.1705 - val\_accuracy: 0.7060  
Epoch 45/100  
2/2 [=====] - 1s 519ms/step - loss: 1.1471 - accuracy: 0.7091 - val\_loss:  
1.1688 - val\_accuracy: 0.7068  
Epoch 46/100  
2/2 [=====] - 1s 538ms/step - loss: 1.1442 - accuracy: 0.7095 - val\_loss:  
1.1669 - val\_accuracy: 0.7068  
Epoch 47/100  
2/2 [=====] - 1s 526ms/step - loss: 1.1402 - accuracy: 0.7100 - val\_loss:  
1.1589 - val\_accuracy: 0.7078  
Epoch 48/100  
2/2 [=====] - 1s 516ms/step - loss: 1.1368 - accuracy: 0.7109 - val\_loss:  
1.1562 - val\_accuracy: 0.7085  
Epoch 49/100  
2/2 [=====] - 1s 522ms/step - loss: 1.1391 - accuracy: 0.7111 - val\_loss:  
1.1613 - val\_accuracy: 0.7088  
Epoch 50/100  
2/2 [=====] - 1s 534ms/step - loss: 1.1412 - accuracy: 0.7105 - val\_loss:  
1.1588 - val\_accuracy: 0.7085  
Epoch 51/100  
2/2 [=====] - 1s 507ms/step - loss: 1.1358 - accuracy: 0.7111 - val\_loss:  
1.1534 - val\_accuracy: 0.7095  
Epoch 52/100  
2/2 [=====] - 1s 505ms/step - loss: 1.1301 - accuracy: 0.7115 - val\_loss:  
1.1408 - val\_accuracy: 0.7108  
Epoch 53/100  
2/2 [=====] - 1s 523ms/step - loss: 1.1262 - accuracy: 0.7127 - val\_loss:  
1.1561 - val\_accuracy: 0.7111  
Epoch 54/100  
2/2 [=====] - 1s 529ms/step - loss: 1.1395 - accuracy: 0.7119 - val\_loss:  
1.1544 - val\_accuracy: 0.7128  
Epoch 55/100  
2/2 [=====] - 1s 514ms/step - loss: 1.1348 - accuracy: 0.7127 - val\_loss:  
1.1467 - val\_accuracy: 0.7123  
Epoch 56/100  
2/2 [=====] - 1s 510ms/step - loss: 1.1230 - accuracy: 0.7138 - val\_loss:  
1.1445 - val\_accuracy: 0.7151  
Epoch 57/100  
2/2 [=====] - 1s 514ms/step - loss: 1.1207 - accuracy: 0.7148 - val\_loss:  
1.1324 - val\_accuracy: 0.7161  
Epoch 58/100  
2/2 [=====] - 1s 520ms/step - loss: 1.1154 - accuracy: 0.7150 - val\_loss:  
1.1299 - val\_accuracy: 0.7156  
Epoch 59/100  
2/2 [=====] - 1s 524ms/step - loss: 1.1111 - accuracy: 0.7150 - val\_loss:  
1.1221 - val\_accuracy: 0.7148  
Epoch 60/100  
2/2 [=====] - 1s 538ms/step - loss: 1.1089 - accuracy: 0.7159 - val\_loss:  
1.1962 - val\_accuracy: 0.7103  
Epoch 61/100  
2/2 [=====] - 1s 523ms/step - loss: 1.1592 - accuracy: 0.7129 - val\_loss:  
1.1504 - val\_accuracy: 0.7095  
Epoch 62/100  
2/2 [=====] - 1s 515ms/step - loss: 1.1424 - accuracy: 0.7111 - val\_loss:  
1.1665 - val\_accuracy: 0.7088  
Epoch 63/100  
2/2 [=====] - 1s 526ms/step - loss: 1.1507 - accuracy: 0.7107 - val\_loss:  
1.1584 - val\_accuracy: 0.7095  
Epoch 64/100  
2/2 [=====] - 1s 527ms/step - loss: 1.1358 - accuracy: 0.7113 - val\_loss:  
1.1567 - val\_accuracy: 0.7106  
Epoch 65/100  
2/2 [=====] - 1s 517ms/step - loss: 1.1334 - accuracy: 0.7113 - val\_loss:  
1.1583 - val\_accuracy: 0.7111  
Epoch 66/100

Epoch 66/100  
2/2 [=====] - 1s 523ms/step - loss: 1.1294 - accuracy: 0.7115 - val\_loss:  
1.1582 - val\_accuracy: 0.7085  
Epoch 67/100  
2/2 [=====] - 1s 521ms/step - loss: 1.1237 - accuracy: 0.7125 - val\_loss:  
1.1463 - val\_accuracy: 0.7098  
Epoch 68/100  
2/2 [=====] - 1s 536ms/step - loss: 1.1179 - accuracy: 0.7128 - val\_loss:  
1.1357 - val\_accuracy: 0.7106  
Epoch 69/100  
2/2 [=====] - 1s 520ms/step - loss: 1.1146 - accuracy: 0.7135 - val\_loss:  
1.1309 - val\_accuracy: 0.7108  
Epoch 70/100  
2/2 [=====] - 1s 530ms/step - loss: 1.1109 - accuracy: 0.7141 - val\_loss:  
1.1264 - val\_accuracy: 0.7133  
Epoch 71/100  
2/2 [=====] - 1s 523ms/step - loss: 1.1043 - accuracy: 0.7149 - val\_loss:  
1.1223 - val\_accuracy: 0.7141  
Epoch 72/100  
2/2 [=====] - 1s 535ms/step - loss: 1.1000 - accuracy: 0.7159 - val\_loss:  
1.1154 - val\_accuracy: 0.7161  
Epoch 73/100  
2/2 [=====] - 1s 521ms/step - loss: 1.0958 - accuracy: 0.7165 - val\_loss:  
1.1061 - val\_accuracy: 0.7163  
Epoch 74/100  
2/2 [=====] - 1s 517ms/step - loss: 1.0912 - accuracy: 0.7175 - val\_loss:  
1.0990 - val\_accuracy: 0.7166  
Epoch 75/100  
2/2 [=====] - 1s 531ms/step - loss: 1.0868 - accuracy: 0.7179 - val\_loss:  
1.0958 - val\_accuracy: 0.7176  
Epoch 76/100  
2/2 [=====] - 1s 534ms/step - loss: 1.0840 - accuracy: 0.7185 - val\_loss:  
1.0942 - val\_accuracy: 0.7188  
Epoch 77/100  
2/2 [=====] - 1s 529ms/step - loss: 1.0807 - accuracy: 0.7187 - val\_loss:  
1.0966 - val\_accuracy: 0.7196  
Epoch 78/100  
2/2 [=====] - 1s 522ms/step - loss: 1.0800 - accuracy: 0.7190 - val\_loss:  
1.0965 - val\_accuracy: 0.7204  
Epoch 79/100  
2/2 [=====] - 1s 510ms/step - loss: 1.0751 - accuracy: 0.7196 - val\_loss:  
1.0854 - val\_accuracy: 0.7211  
Epoch 80/100  
2/2 [=====] - 1s 526ms/step - loss: 1.0770 - accuracy: 0.7197 - val\_loss:  
1.0805 - val\_accuracy: 0.7224  
Epoch 81/100  
2/2 [=====] - 1s 528ms/step - loss: 1.0700 - accuracy: 0.7205 - val\_loss:  
1.0762 - val\_accuracy: 0.7214  
Epoch 82/100  
2/2 [=====] - 1s 516ms/step - loss: 1.0692 - accuracy: 0.7207 - val\_loss:  
1.0738 - val\_accuracy: 0.7214  
Epoch 83/100  
2/2 [=====] - 1s 520ms/step - loss: 1.0663 - accuracy: 0.7210 - val\_loss:  
1.0813 - val\_accuracy: 0.7224  
Epoch 84/100  
2/2 [=====] - 1s 531ms/step - loss: 1.0648 - accuracy: 0.7208 - val\_loss:  
1.0679 - val\_accuracy: 0.7219  
Epoch 85/100  
2/2 [=====] - 1s 526ms/step - loss: 1.0623 - accuracy: 0.7214 - val\_loss:  
1.0720 - val\_accuracy: 0.7216  
Epoch 86/100  
2/2 [=====] - 1s 498ms/step - loss: 1.0605 - accuracy: 0.7218 - val\_loss:  
1.0614 - val\_accuracy: 0.7231  
Epoch 87/100  
2/2 [=====] - 1s 523ms/step - loss: 1.0569 - accuracy: 0.7218 - val\_loss:  
1.0646 - val\_accuracy: 0.7229  
Epoch 88/100  
2/2 [=====] - 1s 544ms/step - loss: 1.0554 - accuracy: 0.7222 - val\_loss:  
1.0564 - val\_accuracy: 0.7236  
Epoch 89/100  
2/2 [=====] - 1s 530ms/step - loss: 1.0518 - accuracy: 0.7224 - val\_loss:  
1.0524 - val\_accuracy: 0.7234  
Epoch 90/100  
2/2 [=====] - 1s 509ms/step - loss: 1.0496 - accuracy: 0.7227 - val\_loss:  
1.0523 - val\_accuracy: 0.7241  
Epoch 91/100  
2/2 [=====] - 1s 508ms/step - loss: 1.0486 - accuracy: 0.7230 - val\_loss:  
1.0538 - val\_accuracy: 0.7226

```

1.0000 - val_accuracy: 0.7229
Epoch 92/100
2/2 [=====] - 1s 529ms/step - loss: 1.0483 - accuracy: 0.7229 - val_loss:
1.0508 - val_accuracy: 0.7236
Epoch 93/100
2/2 [=====] - 1s 523ms/step - loss: 1.0464 - accuracy: 0.7229 - val_loss:
1.0504 - val_accuracy: 0.7234
Epoch 94/100
2/2 [=====] - 1s 518ms/step - loss: 1.0475 - accuracy: 0.7234 - val_loss:
1.0502 - val_accuracy: 0.7231
Epoch 95/100
2/2 [=====] - 1s 515ms/step - loss: 1.0435 - accuracy: 0.7237 - val_loss:
1.0497 - val_accuracy: 0.7229
Epoch 96/100
2/2 [=====] - 1s 531ms/step - loss: 1.0415 - accuracy: 0.7238 - val_loss:
1.0574 - val_accuracy: 0.7246
Epoch 97/100
2/2 [=====] - 1s 525ms/step - loss: 1.0480 - accuracy: 0.7223 - val_loss:
1.0707 - val_accuracy: 0.7231
Epoch 98/100
2/2 [=====] - 1s 519ms/step - loss: 1.0442 - accuracy: 0.7230 - val_loss:
1.0528 - val_accuracy: 0.7216
Epoch 99/100
2/2 [=====] - 1s 529ms/step - loss: 1.0434 - accuracy: 0.7229 - val_loss:
1.0457 - val_accuracy: 0.7244
Epoch 100/100
2/2 [=====] - 1s 530ms/step - loss: 1.0381 - accuracy: 0.7235 - val_loss:
1.0492 - val_accuracy: 0.7256

```

Out[24]:

```
<tensorflow.python.keras.callbacks.History at 0x7f38504b6e90>
```

In [25]:

```

#https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/
from math import log
from numpy import array
from numpy import argmax
import numpy as np
def beam_search_decoder(data, k):
    sequences = [[list(), 0.0]]
    # walk over each step in sequence
    #print(sequences)
    for row in data:
        all_candidates = list()
        # expand each current candidate
        for i in range(len(sequences)):
            seq, score = sequences[i]
            for j in range(len(row)):
                candidate = [seq + [j], score - np.log(row[j])]
                all_candidates.append(candidate)
        # order all candidates by score
        ordered = sorted(all_candidates, key=lambda tup:tup[1])
        sequences = ordered[:k]
    return sequences

```

In [29]:

```

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 x])
    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 for beam==3 : ")
    x=model.predict(source_padded_docs_test[i:i+1])

    res=beam_search_decoder(x[0],3)

    y1=prediction(res[0][0])

```

```
y3=prediction(res[2][0])
y3=y3.split(' ')
y_lst3=[]
for i in y3:
    y_lst3.append(i)
print(' '.join(y_lst3))
print('> *180)
```

[illegible]



[illegible]

Input text:

Er... Yeah i think not. Cøz we dun know which one we got assigned. Ü not sleeping yet? Haha... My hair s  
till wet that's why...

Actual Output:

Yes, I don't think so. Cause we don't know which one we will be assigned. Are you not going to sleep yet? Haha. My hair is still wet, that's why.

Predicted Output for beam==3 :

Er.

i

[illegible]

Er.

ii

[illegible]

Er.

i

[illegible]

Input text:

Well... Izzit true for u?

Actual Output:

Well. Is it true for you?

Predicted Output for beam==3 :

[illegible][illegible][illegible][illegible]

Input text:

mohd sultan's double o.my og goin.but i not close to em.i wana ask fion along lei-if ü on.

Actual Output:

Mohd sultan's double o. My Og going. But I'm not close to them. I want to ask Fion along, if you on.

Predicted Output for beam==3 :

Mohd sultan's doubbe o

○

&lt;PAD&gt;&lt;PAD&gt;

[illegible]

[illegible]

[illegible]

Input text:

Shall i buy tis mambo watch tt cost 80 bucks...

Actual Output:

Shall I buy this Mambo watch that costs 80 bucks?

Predicted Output for beam==3 :

[illegible][illegible][illegible][illegible]

Input text:

MY NEW YEARS EVE WAS OK. I WENT TO A PARTY WITH MY BOYFRIEND. WHO IS THIS SI THEN HEY

Actual Output:

My new year evening was ok. I went to a party with my boyfriend. Who is this?

Predicted Output for beam==3 :

[illegible][illegible][illegible][illegible]

Input text:

Haha..Cause supervisor go oversea lo.Then no one look after me lo..Hehe.But i still got find thing do lo.

Actual Output:

Haha. Because supervisor went overseas. Then no one looks after me. Hehe. But I still find things to do.

Predicted Output for beam==3 :

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Input text:

Hey yun ask you ah... where did you the answers for the past year exam papers from?

Actual Output:

Hey Yun, can I ask you? Where did you get the answers for the past year exam papers from?

Predicted Output for beam==3 :

[illegible]

Input text:

Happy Valentine's Day... May this day of yours be blessed with happiness n laughter... Good day ahead.

Actual Output:

Happy Valentine's Day. May this day of yours be blessed with happiness and laughter. Good day ahead.

Predicted Output for beam==3 :

[illegible]

[illegible]

In [30]:

```

import nltk.translate.bleu_score as bleu
bleu_score1=[]
bleu_score2=[]
bleu_score3=[]

for i in range(20):
    b=list(y_test[i:i+1])
    x=model.predict(source_padded_docs_test[i:i+1])

    res=beam_search_decoder(x[0],3)

    y1=prediction(res[0][0])
    y1=y1.split(' ')
    y_lst1=[]
    for i in y1:
        if '<' in i:
            continue
        else:
            y_lst1.append(i)
    bleu_score1.append(bleu.sentence_bleu([b[0].split()],y_lst1))

    y2=prediction(res[1][0])
    y2=y2.split(' ')
    y_lst2=[]
    for i in y2:
        if i=='<PAD>':
            continue
        else:
            y_lst2.append(i)
    bleu_score2.append(bleu.sentence_bleu([b[0].split()],y_lst2))

    y3=prediction(res[2][0])
    y3=y3.split(' ')
    y_lst3=[]
    for i in y3:
        if i=='<PAD>':
            continue
        else:
            y_lst3.append(i)
    bleu_score3.append(bleu.sentence_bleu([b[0].split()],y_lst3))

print("The Average Bleu Score1 is: ",sum(bleu_score1)/20)
print('>'*180)
print("The Average Bleu Score2 is: ",sum(bleu_score2)/20)
print('>'*180)
print("The Average Bleu Score3 is: ",sum(bleu_score3)/20)
print('>'*180)

```

```
/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)
```

The Average Bleu Score1 is: 0.2746536825395667

[illegible]

The Average Bleu Score2 is: 0.2718117637320006

[illegible]

The Average Bleu Score3 is: 0.273708969340681

[illegible]

Model2:

In [35]:

```
input=tf.keras.layers.Input(shape=(199,))
embed=tf.keras.layers.Embedding(source_vocab_size,256, input_length=source_padded_docs_train.shape[1])(input)
lstm1=tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(128, return_sequences=True))(embed)
dense=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(512,activation='relu'))(lstm1)
drop=tf.keras.layers.Dropout(0.5)(dense)
```



```

output=tf.keras.layers.TimeDistributed(tf.keras.layers.Dense(target_vocab_size, activation='softmax')) (dr
model=tf.keras.models.Model(inputs=input,outputs=output)
model.summary()

```

Model: "model\_2"

| Layer (type)                 | Output Shape     | Param # |
|------------------------------|------------------|---------|
| =====                        |                  |         |
| input_3 (InputLayer)         | [(None, 199)]    | 0       |
| embedding_2 (Embedding)      | (None, 199, 256) | 26624   |
| bidirectional_1 (Bidirection | (None, 199, 256) | 394240  |
| time_distributed_4 (TimeDist | (None, 199, 512) | 131584  |
| dropout_2 (Dropout)          | (None, 199, 512) | 0       |
| time_distributed_5 (TimeDist | (None, 199, 91)  | 46683   |
| =====                        |                  |         |
| Total params: 599,131        |                  |         |
| Trainable params: 599,131    |                  |         |
| Non-trainable params: 0      |                  |         |
| =====                        |                  |         |

In [36]:

```

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

```

In [37]:

```

model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=100,
          validation_data=(source_padded_docs_test,target_padded_docs_test))

```

```

Epoch 1/100
2/2 [=====] - 6s 2s/step - loss: 3.5634 - accuracy: 0.3871 - val_loss: 5.6043 -
val_accuracy: 0.6420
Epoch 2/100
2/2 [=====] - 2s 795ms/step - loss: 4.6147 - accuracy: 0.6449 - val_loss:
1.5023 - val_accuracy: 0.6681
Epoch 3/100
2/2 [=====] - 2s 846ms/step - loss: 1.5370 - accuracy: 0.6425 - val_loss:
1.6905 - val_accuracy: 0.6917
Epoch 4/100
2/2 [=====] - 2s 814ms/step - loss: 1.7233 - accuracy: 0.6717 - val_loss:
1.5941 - val_accuracy: 0.6683
Epoch 5/100
2/2 [=====] - 2s 814ms/step - loss: 1.5887 - accuracy: 0.6811 - val_loss:
1.5098 - val_accuracy: 0.6922
Epoch 6/100
2/2 [=====] - 2s 809ms/step - loss: 1.5349 - accuracy: 0.6805 - val_loss:
1.4723 - val_accuracy: 0.6578
Epoch 7/100
2/2 [=====] - 2s 815ms/step - loss: 1.4904 - accuracy: 0.6627 - val_loss:
1.4265 - val_accuracy: 0.6714
Epoch 8/100
2/2 [=====] - 2s 827ms/step - loss: 1.4400 - accuracy: 0.6833 - val_loss:
1.3469 - val_accuracy: 0.6907
Epoch 9/100
2/2 [=====] - 2s 808ms/step - loss: 1.3753 - accuracy: 0.6936 - val_loss:
1.3098 - val_accuracy: 0.6950
Epoch 10/100
2/2 [=====] - 2s 808ms/step - loss: 1.2997 - accuracy: 0.6980 - val_loss:
1.2679 - val_accuracy: 0.6995
Epoch 11/100
2/2 [=====] - 2s 804ms/step - loss: 1.2868 - accuracy: 0.6919 - val_loss:
1.2738 - val_accuracy: 0.6972
Epoch 12/100
2/2 [=====] - 2s 846ms/step - loss: 1.2676 - accuracy: 0.6973 - val_loss:
1.2560 - val_accuracy: 0.6965
Epoch 13/100
2/2 [=====] - 2s 791ms/step - loss: 1.2517 - accuracy: 0.6989 - val_loss:
1.2668 - val_accuracy: 0.6965
Epoch 14/100
2/2 [=====] - 2s 824ms/step - loss: 1.2433 - accuracy: 0.7001 - val_loss:
1.2634 - val_accuracy: 0.6980
Epoch 15/100
2/2 [=====] - 2s 790ms/step - loss: 1.2346 - accuracy: 0.7009 - val_loss:
1.2386 - val accuracy: 0.6980

```

Epoch 16/100  
2/2 [=====] - 2s 799ms/step - loss: 1.2265 - accuracy: 0.7012 - val\_loss: 1.2164 - val\_accuracy: 0.6982  
Epoch 17/100  
2/2 [=====] - 2s 804ms/step - loss: 1.2222 - accuracy: 0.7021 - val\_loss: 1.2137 - val\_accuracy: 0.6997  
Epoch 18/100  
2/2 [=====] - 2s 808ms/step - loss: 1.2208 - accuracy: 0.7020 - val\_loss: 1.2102 - val\_accuracy: 0.6990  
Epoch 19/100  
2/2 [=====] - 2s 821ms/step - loss: 1.2168 - accuracy: 0.7020 - val\_loss: 1.2107 - val\_accuracy: 0.7000  
Epoch 20/100  
2/2 [=====] - 2s 798ms/step - loss: 1.2131 - accuracy: 0.7021 - val\_loss: 1.2113 - val\_accuracy: 0.7000  
Epoch 21/100  
2/2 [=====] - 2s 806ms/step - loss: 1.2103 - accuracy: 0.7022 - val\_loss: 1.2118 - val\_accuracy: 0.6997  
Epoch 22/100  
2/2 [=====] - 2s 782ms/step - loss: 1.2084 - accuracy: 0.7022 - val\_loss: 1.2085 - val\_accuracy: 0.7000  
Epoch 23/100  
2/2 [=====] - 2s 814ms/step - loss: 1.2070 - accuracy: 0.7023 - val\_loss: 1.2047 - val\_accuracy: 0.6997  
Epoch 24/100  
2/2 [=====] - 2s 812ms/step - loss: 1.2046 - accuracy: 0.7023 - val\_loss: 1.2054 - val\_accuracy: 0.7000  
Epoch 25/100  
2/2 [=====] - 2s 792ms/step - loss: 1.2022 - accuracy: 0.7025 - val\_loss: 1.2049 - val\_accuracy: 0.7005  
Epoch 26/100  
2/2 [=====] - 2s 810ms/step - loss: 1.2005 - accuracy: 0.7025 - val\_loss: 1.2027 - val\_accuracy: 0.7005  
Epoch 27/100  
2/2 [=====] - 2s 801ms/step - loss: 1.1985 - accuracy: 0.7025 - val\_loss: 1.1986 - val\_accuracy: 0.7003  
Epoch 28/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1968 - accuracy: 0.7026 - val\_loss: 1.1960 - val\_accuracy: 0.7010  
Epoch 29/100  
2/2 [=====] - 2s 809ms/step - loss: 1.1951 - accuracy: 0.7027 - val\_loss: 1.1953 - val\_accuracy: 0.7010  
Epoch 30/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1934 - accuracy: 0.7028 - val\_loss: 1.1932 - val\_accuracy: 0.7010  
Epoch 31/100  
2/2 [=====] - 2s 792ms/step - loss: 1.1919 - accuracy: 0.7029 - val\_loss: 1.1916 - val\_accuracy: 0.7010  
Epoch 32/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1906 - accuracy: 0.7029 - val\_loss: 1.1919 - val\_accuracy: 0.7010  
Epoch 33/100  
2/2 [=====] - 2s 799ms/step - loss: 1.1905 - accuracy: 0.7030 - val\_loss: 1.1916 - val\_accuracy: 0.7010  
Epoch 34/100  
2/2 [=====] - 2s 818ms/step - loss: 1.1895 - accuracy: 0.7030 - val\_loss: 1.1867 - val\_accuracy: 0.7008  
Epoch 35/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1933 - accuracy: 0.7031 - val\_loss: 1.1998 - val\_accuracy: 0.7018  
Epoch 36/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1950 - accuracy: 0.7032 - val\_loss: 1.1964 - val\_accuracy: 0.7015  
Epoch 37/100  
2/2 [=====] - 2s 799ms/step - loss: 1.1955 - accuracy: 0.7031 - val\_loss: 1.1936 - val\_accuracy: 0.7008  
Epoch 38/100  
2/2 [=====] - 2s 795ms/step - loss: 1.1905 - accuracy: 0.7034 - val\_loss: 1.1893 - val\_accuracy: 0.7010  
Epoch 39/100  
2/2 [=====] - 2s 809ms/step - loss: 1.1872 - accuracy: 0.7035 - val\_loss: 1.1875 - val\_accuracy: 0.7013  
Epoch 40/100  
2/2 [=====] - 2s 819ms/step - loss: 1.1856 - accuracy: 0.7037 - val\_loss: 1.1873 - val\_accuracy: 0.7018  
Epoch 41/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1836 - accuracy: 0.7038 - val\_loss:

1.1944 - val\_accuracy: 0.7020  
Epoch 42/100  
2/2 [=====] - 2s 822ms/step - loss: 1.1844 - accuracy: 0.7040 - val\_loss:  
1.1823 - val\_accuracy: 0.7023  
Epoch 43/100  
2/2 [=====] - 2s 793ms/step - loss: 1.1821 - accuracy: 0.7042 - val\_loss:  
1.1813 - val\_accuracy: 0.7025  
Epoch 44/100  
2/2 [=====] - 2s 821ms/step - loss: 1.1797 - accuracy: 0.7045 - val\_loss:  
1.1855 - val\_accuracy: 0.7030  
Epoch 45/100  
2/2 [=====] - 2s 799ms/step - loss: 1.1834 - accuracy: 0.7047 - val\_loss:  
1.1832 - val\_accuracy: 0.7020  
Epoch 46/100  
2/2 [=====] - 2s 817ms/step - loss: 1.1771 - accuracy: 0.7049 - val\_loss:  
1.1794 - val\_accuracy: 0.7025  
Epoch 47/100  
2/2 [=====] - 2s 811ms/step - loss: 1.1768 - accuracy: 0.7050 - val\_loss:  
1.1791 - val\_accuracy: 0.7023  
Epoch 48/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1751 - accuracy: 0.7051 - val\_loss:  
1.1778 - val\_accuracy: 0.7023  
Epoch 49/100  
2/2 [=====] - 2s 804ms/step - loss: 1.1731 - accuracy: 0.7052 - val\_loss:  
1.1776 - val\_accuracy: 0.7015  
Epoch 50/100  
2/2 [=====] - 2s 798ms/step - loss: 1.1713 - accuracy: 0.7054 - val\_loss:  
1.1806 - val\_accuracy: 0.7025  
Epoch 51/100  
2/2 [=====] - 2s 809ms/step - loss: 1.1691 - accuracy: 0.7058 - val\_loss:  
1.1726 - val\_accuracy: 0.7023  
Epoch 52/100  
2/2 [=====] - 2s 813ms/step - loss: 1.1677 - accuracy: 0.7058 - val\_loss:  
1.1676 - val\_accuracy: 0.7023  
Epoch 53/100  
2/2 [=====] - 2s 806ms/step - loss: 1.1630 - accuracy: 0.7060 - val\_loss:  
1.1713 - val\_accuracy: 0.7033  
Epoch 54/100  
2/2 [=====] - 2s 811ms/step - loss: 1.1619 - accuracy: 0.7060 - val\_loss:  
1.1691 - val\_accuracy: 0.7028  
Epoch 55/100  
2/2 [=====] - 2s 809ms/step - loss: 1.1592 - accuracy: 0.7062 - val\_loss:  
1.1632 - val\_accuracy: 0.7028  
Epoch 56/100  
2/2 [=====] - 2s 810ms/step - loss: 1.1735 - accuracy: 0.7062 - val\_loss:  
1.2816 - val\_accuracy: 0.6997  
Epoch 57/100  
2/2 [=====] - 2s 790ms/step - loss: 1.2281 - accuracy: 0.7028 - val\_loss:  
1.1937 - val\_accuracy: 0.7045  
Epoch 58/100  
2/2 [=====] - 2s 804ms/step - loss: 1.2109 - accuracy: 0.7060 - val\_loss:  
1.2041 - val\_accuracy: 0.7043  
Epoch 59/100  
2/2 [=====] - 2s 798ms/step - loss: 1.1891 - accuracy: 0.7063 - val\_loss:  
1.1819 - val\_accuracy: 0.7033  
Epoch 60/100  
2/2 [=====] - 2s 789ms/step - loss: 1.1886 - accuracy: 0.7060 - val\_loss:  
1.1933 - val\_accuracy: 0.7033  
Epoch 61/100  
2/2 [=====] - 2s 810ms/step - loss: 1.1780 - accuracy: 0.7063 - val\_loss:  
1.2038 - val\_accuracy: 0.7043  
Epoch 62/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1799 - accuracy: 0.7068 - val\_loss:  
1.2012 - val\_accuracy: 0.7048  
Epoch 63/100  
2/2 [=====] - 2s 820ms/step - loss: 1.1737 - accuracy: 0.7072 - val\_loss:  
1.1672 - val\_accuracy: 0.7053  
Epoch 64/100  
2/2 [=====] - 2s 801ms/step - loss: 1.1673 - accuracy: 0.7072 - val\_loss:  
1.1689 - val\_accuracy: 0.7055  
Epoch 65/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1679 - accuracy: 0.7070 - val\_loss:  
1.1591 - val\_accuracy: 0.7065  
Epoch 66/100  
2/2 [=====] - 2s 812ms/step - loss: 1.1598 - accuracy: 0.7077 - val\_loss:  
1.1642 - val\_accuracy: 0.7060  
Epoch 67/100

Epoch 67/100  
2/2 [=====] - 2s 811ms/step - loss: 1.1596 - accuracy: 0.7080 - val\_loss:  
1.1607 - val\_accuracy: 0.7063  
Epoch 68/100  
2/2 [=====] - 2s 798ms/step - loss: 1.1550 - accuracy: 0.7082 - val\_loss:  
1.1600 - val\_accuracy: 0.7058  
Epoch 69/100  
2/2 [=====] - 2s 799ms/step - loss: 1.1548 - accuracy: 0.7087 - val\_loss:  
1.1552 - val\_accuracy: 0.7075  
Epoch 70/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1515 - accuracy: 0.7088 - val\_loss:  
1.1543 - val\_accuracy: 0.7075  
Epoch 71/100  
2/2 [=====] - 2s 818ms/step - loss: 1.1514 - accuracy: 0.7087 - val\_loss:  
1.1511 - val\_accuracy: 0.7070  
Epoch 72/100  
2/2 [=====] - 2s 776ms/step - loss: 1.1483 - accuracy: 0.7091 - val\_loss:  
1.1502 - val\_accuracy: 0.7075  
Epoch 73/100  
2/2 [=====] - 2s 815ms/step - loss: 1.1465 - accuracy: 0.7095 - val\_loss:  
1.1488 - val\_accuracy: 0.7068  
Epoch 74/100  
2/2 [=====] - 2s 809ms/step - loss: 1.1448 - accuracy: 0.7099 - val\_loss:  
1.1465 - val\_accuracy: 0.7093  
Epoch 75/100  
2/2 [=====] - 2s 812ms/step - loss: 1.1438 - accuracy: 0.7100 - val\_loss:  
1.1481 - val\_accuracy: 0.7068  
Epoch 76/100  
2/2 [=====] - 2s 816ms/step - loss: 1.1435 - accuracy: 0.7099 - val\_loss:  
1.1458 - val\_accuracy: 0.7078  
Epoch 77/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1413 - accuracy: 0.7102 - val\_loss:  
1.1429 - val\_accuracy: 0.7090  
Epoch 78/100  
2/2 [=====] - 2s 815ms/step - loss: 1.1403 - accuracy: 0.7106 - val\_loss:  
1.1432 - val\_accuracy: 0.7080  
Epoch 79/100  
2/2 [=====] - 2s 814ms/step - loss: 1.1381 - accuracy: 0.7107 - val\_loss:  
1.1424 - val\_accuracy: 0.7088  
Epoch 80/100  
2/2 [=====] - 2s 814ms/step - loss: 1.1360 - accuracy: 0.7108 - val\_loss:  
1.1391 - val\_accuracy: 0.7093  
Epoch 81/100  
2/2 [=====] - 2s 815ms/step - loss: 1.1341 - accuracy: 0.7114 - val\_loss:  
1.1369 - val\_accuracy: 0.7093  
Epoch 82/100  
2/2 [=====] - 2s 817ms/step - loss: 1.1324 - accuracy: 0.7114 - val\_loss:  
1.1346 - val\_accuracy: 0.7098  
Epoch 83/100  
2/2 [=====] - 2s 813ms/step - loss: 1.1307 - accuracy: 0.7121 - val\_loss:  
1.1330 - val\_accuracy: 0.7101  
Epoch 84/100  
2/2 [=====] - 2s 803ms/step - loss: 1.1290 - accuracy: 0.7122 - val\_loss:  
1.1306 - val\_accuracy: 0.7103  
Epoch 85/100  
2/2 [=====] - 2s 813ms/step - loss: 1.1276 - accuracy: 0.7126 - val\_loss:  
1.1286 - val\_accuracy: 0.7113  
Epoch 86/100  
2/2 [=====] - 2s 812ms/step - loss: 1.1259 - accuracy: 0.7129 - val\_loss:  
1.1354 - val\_accuracy: 0.7095  
Epoch 87/100  
2/2 [=====] - 2s 819ms/step - loss: 1.1286 - accuracy: 0.7127 - val\_loss:  
1.1227 - val\_accuracy: 0.7123  
Epoch 88/100  
2/2 [=====] - 2s 810ms/step - loss: 1.1265 - accuracy: 0.7133 - val\_loss:  
1.1204 - val\_accuracy: 0.7108  
Epoch 89/100  
2/2 [=====] - 2s 805ms/step - loss: 1.1231 - accuracy: 0.7133 - val\_loss:  
1.1194 - val\_accuracy: 0.7123  
Epoch 90/100  
2/2 [=====] - 2s 815ms/step - loss: 1.1217 - accuracy: 0.7138 - val\_loss:  
1.1196 - val\_accuracy: 0.7126  
Epoch 91/100  
2/2 [=====] - 2s 807ms/step - loss: 1.1194 - accuracy: 0.7137 - val\_loss:  
1.1165 - val\_accuracy: 0.7133  
Epoch 92/100  
2/2 [=====] - 2s 834ms/step - loss: 1.1208 - accuracy: 0.7139 - val\_loss:  
1.1387 - val\_accuracy: 0.7070

```
1.1253 - val_accuracy: 0.7108
Epoch 93/100
2/2 [=====] - 2s 802ms/step - loss: 1.1285 - accuracy: 0.7117 - val_loss:
1.1253 - val_accuracy: 0.7108
Epoch 94/100
2/2 [=====] - 2s 808ms/step - loss: 1.1232 - accuracy: 0.7134 - val_loss:
1.1156 - val_accuracy: 0.7143
Epoch 95/100
2/2 [=====] - 2s 823ms/step - loss: 1.1211 - accuracy: 0.7146 - val_loss:
1.1146 - val_accuracy: 0.7141
Epoch 96/100
2/2 [=====] - 2s 800ms/step - loss: 1.1158 - accuracy: 0.7148 - val_loss:
1.1170 - val_accuracy: 0.7136
Epoch 97/100
2/2 [=====] - 2s 798ms/step - loss: 1.1153 - accuracy: 0.7153 - val_loss:
1.1118 - val_accuracy: 0.7138
Epoch 98/100
2/2 [=====] - 2s 814ms/step - loss: 1.1125 - accuracy: 0.7158 - val_loss:
1.1088 - val_accuracy: 0.7148
Epoch 99/100
2/2 [=====] - 2s 809ms/step - loss: 1.1100 - accuracy: 0.7156 - val_loss:
1.1064 - val_accuracy: 0.7151
Epoch 100/100
2/2 [=====] - 2s 799ms/step - loss: 1.1086 - accuracy: 0.7160 - val_loss:
1.1010 - val_accuracy: 0.7161
```

Out[37]:

```
<tensorflow.python.keras.callbacks.History at 0x7f37e7afbe50>
```

In [38]:

```
model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=20,
          validation_data=(source_padded_docs_test,target_padded_docs_test))
```

```

Epoch 1/20
2/2 [=====] - 2s 824ms/step - loss: 1.1064 - accuracy: 0.7168 - val_loss:
1.1013 - val_accuracy: 0.7168
Epoch 2/20
2/2 [=====] - 2s 811ms/step - loss: 1.1049 - accuracy: 0.7171 - val_loss:
1.1004 - val_accuracy: 0.7166
Epoch 3/20
2/2 [=====] - 2s 811ms/step - loss: 1.1036 - accuracy: 0.7175 - val_loss:
1.0921 - val_accuracy: 0.7173
Epoch 4/20
2/2 [=====] - 2s 810ms/step - loss: 1.1025 - accuracy: 0.7183 - val_loss:
1.0914 - val_accuracy: 0.7173
Epoch 5/20
2/2 [=====] - 2s 814ms/step - loss: 1.0998 - accuracy: 0.7179 - val_loss:
1.0927 - val_accuracy: 0.7163
Epoch 6/20
2/2 [=====] - 2s 824ms/step - loss: 1.0994 - accuracy: 0.7181 - val_loss:
1.0879 - val_accuracy: 0.7186
Epoch 7/20
2/2 [=====] - 2s 811ms/step - loss: 1.0962 - accuracy: 0.7191 - val_loss:
1.0864 - val_accuracy: 0.7168
Epoch 8/20
2/2 [=====] - 2s 816ms/step - loss: 1.0962 - accuracy: 0.7187 - val_loss:
1.0852 - val_accuracy: 0.7188
Epoch 9/20
2/2 [=====] - 2s 776ms/step - loss: 1.0934 - accuracy: 0.7193 - val_loss:
1.0815 - val_accuracy: 0.7183
Epoch 10/20
2/2 [=====] - 2s 810ms/step - loss: 1.0914 - accuracy: 0.7202 - val_loss:
1.0808 - val_accuracy: 0.7186
Epoch 11/20
2/2 [=====] - 2s 799ms/step - loss: 1.0919 - accuracy: 0.7193 - val_loss:
1.0788 - val_accuracy: 0.7188
Epoch 12/20
2/2 [=====] - 2s 825ms/step - loss: 1.0906 - accuracy: 0.7202 - val_loss:
1.0778 - val_accuracy: 0.7204
Epoch 13/20
2/2 [=====] - 2s 822ms/step - loss: 1.0877 - accuracy: 0.7209 - val_loss:
1.0755 - val_accuracy: 0.7183
Epoch 14/20
2/2 [=====] - 2s 806ms/step - loss: 1.0915 - accuracy: 0.7198 - val_loss:
1.0748 - val_accuracy: 0.7206
Epoch 15/20
2/2 [=====] - 2s 811ms/step - loss: 1.0898 - accuracy: 0.7201 - val_loss:
1.0748 - val_accuracy: 0.7211
Epoch 16/20
2/2 [=====] - 2s 798ms/step - loss: 1.0848 - accuracy: 0.7210 - val_loss:
1.0745 - val_accuracy: 0.7221
Epoch 17/20
2/2 [=====] - 2s 800ms/step - loss: 1.0831 - accuracy: 0.7210 - val_loss:
1.0731 - val_accuracy: 0.7219
Epoch 18/20
2/2 [=====] - 2s 801ms/step - loss: 1.0810 - accuracy: 0.7216 - val_loss:
1.0750 - val_accuracy: 0.7221
Epoch 19/20
2/2 [=====] - 2s 816ms/step - loss: 1.0800 - accuracy: 0.7219 - val_loss:
1.0666 - val_accuracy: 0.7219
Epoch 20/20
2/2 [=====] - 2s 800ms/step - loss: 1.0786 - accuracy: 0.7225 - val_loss:
1.0725 - val_accuracy: 0.7198

```

Out[38]:

```
<tensorflow.python.keras.callbacks.History at 0x7f37e5149e90>
```

In [39]:

```

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 [=====] - 2s 835ms/step - loss: 1.0831 - accuracy: 0.7205 - val_loss:
1.0790 - val_accuracy: 0.7206
Epoch 2/50
2/2 [=====] - 2s 829ms/step - loss: 1.0825 - accuracy: 0.7215 - val_loss:
1.0709 - val_accuracy: 0.7206
Epoch 3/50
2/2 [=====] - 2s 823ms/step - loss: 1.0784 - accuracy: 0.7212 - val_loss:
1.0703 - val_accuracy: 0.7209
Epoch 4/50
2/2 [=====] - 2s 822ms/step - loss: 1.0772 - accuracy: 0.7223 - val_loss:

```

1.0614 - val\_accuracy: 0.7239  
Epoch 5/50  
2/2 [=====] - 2s 819ms/step - loss: 1.0742 - accuracy: 0.7226 - val\_loss:  
1.0633 - val\_accuracy: 0.7229  
Epoch 6/50  
2/2 [=====] - 2s 789ms/step - loss: 1.0724 - accuracy: 0.7229 - val\_loss:  
1.0633 - val\_accuracy: 0.7231  
Epoch 7/50  
2/2 [=====] - 2s 799ms/step - loss: 1.0703 - accuracy: 0.7237 - val\_loss:  
1.0605 - val\_accuracy: 0.7254  
Epoch 8/50  
2/2 [=====] - 2s 804ms/step - loss: 1.0694 - accuracy: 0.7236 - val\_loss:  
1.0563 - val\_accuracy: 0.7239  
Epoch 9/50  
2/2 [=====] - 2s 829ms/step - loss: 1.0704 - accuracy: 0.7226 - val\_loss:  
1.0687 - val\_accuracy: 0.7231  
Epoch 10/50  
2/2 [=====] - 2s 820ms/step - loss: 1.0709 - accuracy: 0.7230 - val\_loss:  
1.0562 - val\_accuracy: 0.7241  
Epoch 11/50  
2/2 [=====] - 2s 806ms/step - loss: 1.0674 - accuracy: 0.7241 - val\_loss:  
1.0584 - val\_accuracy: 0.7234  
Epoch 12/50  
2/2 [=====] - 2s 813ms/step - loss: 1.0647 - accuracy: 0.7239 - val\_loss:  
1.0529 - val\_accuracy: 0.7264  
Epoch 13/50  
2/2 [=====] - 2s 784ms/step - loss: 1.0625 - accuracy: 0.7247 - val\_loss:  
1.0489 - val\_accuracy: 0.7256  
Epoch 14/50  
2/2 [=====] - 2s 827ms/step - loss: 1.0604 - accuracy: 0.7243 - val\_loss:  
1.0504 - val\_accuracy: 0.7246  
Epoch 15/50  
2/2 [=====] - 2s 806ms/step - loss: 1.0592 - accuracy: 0.7248 - val\_loss:  
1.0488 - val\_accuracy: 0.7246  
Epoch 16/50  
2/2 [=====] - 2s 811ms/step - loss: 1.0598 - accuracy: 0.7250 - val\_loss:  
1.0497 - val\_accuracy: 0.7251  
Epoch 17/50  
2/2 [=====] - 2s 810ms/step - loss: 1.0564 - accuracy: 0.7254 - val\_loss:  
1.0455 - val\_accuracy: 0.7261  
Epoch 18/50  
2/2 [=====] - 2s 812ms/step - loss: 1.0589 - accuracy: 0.7254 - val\_loss:  
1.0728 - val\_accuracy: 0.7176  
Epoch 19/50  
2/2 [=====] - 2s 806ms/step - loss: 1.0853 - accuracy: 0.7180 - val\_loss:  
1.0778 - val\_accuracy: 0.7163  
Epoch 20/50  
2/2 [=====] - 2s 817ms/step - loss: 1.0781 - accuracy: 0.7178 - val\_loss:  
1.0760 - val\_accuracy: 0.7153  
Epoch 21/50  
2/2 [=====] - 2s 812ms/step - loss: 1.0746 - accuracy: 0.7191 - val\_loss:  
1.0707 - val\_accuracy: 0.7171  
Epoch 22/50  
2/2 [=====] - 2s 815ms/step - loss: 1.0699 - accuracy: 0.7209 - val\_loss:  
1.0686 - val\_accuracy: 0.7209  
Epoch 23/50  
2/2 [=====] - 2s 798ms/step - loss: 1.0667 - accuracy: 0.7217 - val\_loss:  
1.0574 - val\_accuracy: 0.7171  
Epoch 24/50  
2/2 [=====] - 2s 808ms/step - loss: 1.0621 - accuracy: 0.7229 - val\_loss:  
1.0517 - val\_accuracy: 0.7206  
Epoch 25/50  
2/2 [=====] - 2s 816ms/step - loss: 1.0588 - accuracy: 0.7233 - val\_loss:  
1.0496 - val\_accuracy: 0.7219  
Epoch 26/50  
2/2 [=====] - 2s 822ms/step - loss: 1.0572 - accuracy: 0.7233 - val\_loss:  
1.0487 - val\_accuracy: 0.7216  
Epoch 27/50  
2/2 [=====] - 2s 781ms/step - loss: 1.0556 - accuracy: 0.7240 - val\_loss:  
1.0426 - val\_accuracy: 0.7211  
Epoch 28/50  
2/2 [=====] - 2s 811ms/step - loss: 1.0529 - accuracy: 0.7243 - val\_loss:  
1.0413 - val\_accuracy: 0.7214  
Epoch 29/50  
2/2 [=====] - 2s 803ms/step - loss: 1.0517 - accuracy: 0.7245 - val\_loss:  
1.0376 - val\_accuracy: 0.7236  
Epoch 30/50

```

2/2 [=====] - 2s 807ms/step - loss: 1.0489 - accuracy: 0.7248 - val_loss:
1.0374 - val_accuracy: 0.7244
Epoch 31/50
2/2 [=====] - 2s 791ms/step - loss: 1.0473 - accuracy: 0.7254 - val_loss:
1.0330 - val_accuracy: 0.7246
Epoch 32/50
2/2 [=====] - 2s 823ms/step - loss: 1.0449 - accuracy: 0.7256 - val_loss:
1.0330 - val_accuracy: 0.7241
Epoch 33/50
2/2 [=====] - 2s 806ms/step - loss: 1.0454 - accuracy: 0.7256 - val_loss:
1.0298 - val_accuracy: 0.7231
Epoch 34/50
2/2 [=====] - 2s 811ms/step - loss: 1.0421 - accuracy: 0.7259 - val_loss:
1.0304 - val_accuracy: 0.7259
Epoch 35/50
2/2 [=====] - 2s 802ms/step - loss: 1.0404 - accuracy: 0.7257 - val_loss:
1.0317 - val_accuracy: 0.7231
Epoch 36/50
2/2 [=====] - 2s 806ms/step - loss: 1.0410 - accuracy: 0.7257 - val_loss:
1.0283 - val_accuracy: 0.7244
Epoch 37/50
2/2 [=====] - 2s 801ms/step - loss: 1.0385 - accuracy: 0.7262 - val_loss:
1.0249 - val_accuracy: 0.7249
Epoch 38/50
2/2 [=====] - 2s 809ms/step - loss: 1.0387 - accuracy: 0.7263 - val_loss:
1.0281 - val_accuracy: 0.7241
Epoch 39/50
2/2 [=====] - 2s 814ms/step - loss: 1.0374 - accuracy: 0.7264 - val_loss:
1.0308 - val_accuracy: 0.7244
Epoch 40/50
2/2 [=====] - 2s 793ms/step - loss: 1.0378 - accuracy: 0.7267 - val_loss:
1.0231 - val_accuracy: 0.7259
Epoch 41/50
2/2 [=====] - 2s 816ms/step - loss: 1.0334 - accuracy: 0.7268 - val_loss:
1.0213 - val_accuracy: 0.7264
Epoch 42/50
2/2 [=====] - 2s 805ms/step - loss: 1.0320 - accuracy: 0.7269 - val_loss:
1.0216 - val_accuracy: 0.7254
Epoch 43/50
2/2 [=====] - 2s 798ms/step - loss: 1.0303 - accuracy: 0.7270 - val_loss:
1.0194 - val_accuracy: 0.7259
Epoch 44/50
2/2 [=====] - 2s 819ms/step - loss: 1.0289 - accuracy: 0.7275 - val_loss:
1.0232 - val_accuracy: 0.7269
Epoch 45/50
2/2 [=====] - 2s 816ms/step - loss: 1.0278 - accuracy: 0.7275 - val_loss:
1.0191 - val_accuracy: 0.7246
Epoch 46/50
2/2 [=====] - 2s 829ms/step - loss: 1.0256 - accuracy: 0.7280 - val_loss:
1.0181 - val_accuracy: 0.7249
Epoch 47/50
2/2 [=====] - 2s 804ms/step - loss: 1.0240 - accuracy: 0.7281 - val_loss:
1.0209 - val_accuracy: 0.7266
Epoch 48/50
2/2 [=====] - 2s 838ms/step - loss: 1.0249 - accuracy: 0.7280 - val_loss:
1.0239 - val_accuracy: 0.7259
Epoch 49/50
2/2 [=====] - 2s 824ms/step - loss: 1.0248 - accuracy: 0.7272 - val_loss:
1.0176 - val_accuracy: 0.7249
Epoch 50/50
2/2 [=====] - 2s 802ms/step - loss: 1.0299 - accuracy: 0.7271 - val_loss:
1.0192 - val_accuracy: 0.7264

```

Out[39]:

```
<tensorflow.python.keras.callbacks.History at 0x7f37e5133f10>
```

In [40]:

```

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 x])
    return y
for i in range(20):
    print("Input text: ")
    a=list(X_test[i:i+1])
    print(a[0])

```





[illegible]

Input text:  
Er... Yeah i think not. Cøz we dun know which one we got assigned. Ü not sleeping yet? Haha... My hair s  
till wet that's why...

Yes, I don't think so. Cause we don't know which one we will be assigned. Are you not going to sleep yet? Haha. My hair is still wet, that's why.

[illegible][illegible]

Actual Output:  
Well. Is it true for you?

[illegible][illegible][illegible]

[illegible]

Input text:

mohd sultan's double o.my og goin.but i not close to em.i wana ask fion along lei-if ü on.

Actual Output:

Mohd sultan's double o. My Oq going. But I'm not close to them. I want to ask Fion along, if you on.

Predicted Output for beam==3 :

[illegible][illegible][illegible]

Input text:

Hi! devin, I am ric. Where u from?

Actual Output:

Hi! Devin, I am Ric. Where are you from?

Predicted Output for beam==3 :

[illegible][illegible][illegible]

Input text:

Save 5 seats can? Try try

Actual Output:

Save 5 seats, can you? Try try.

Predicted Output for beam==3 :

[illegible]

[illegible]



[illegible]

Input text:

Eh. I'm still at the bus stop... Missed the bus. So i might be later than you

Actual Output:

I'm still at the bus stop. I missed the bus. So I might be later than you.

Predicted Output for beam==3 :

[illegible][illegible][illegible][illegible]

Input text:

pj.ur a malay/ chinese,rin?

Actual Output:

PJ. You're a Malay or Chinese, Rin?

Predicted Output for beam==3 :

[illegible][illegible][illegible][illegible]

Input text:

Yar then can say hi... Then later can go for dinner... He like doing project...

Actual Output:

Yes. Can say hi and then go for dinner later. He likes doing project.

Predicted Output for beam==3 :

[illegible]

[illegible]





◀ ▶

```
import nltk.translate.bleu_score as bleu
bleu_score1=[]
bleu_score2=[]
bleu_score3=[]

for i in range(20):
    b=list(y_test[i:i+1])
    x=model.predict(source_padded_docs_test[i:i+1])
    res=beam_search_decoder(x[0],3)

    y1=prediction(res[0][0])
    y1=y1.split(' ')
    y_lst1=[]
    for i in y1:
        if '<' in i:
            continue
        else:
            y_lst1.append(i)
    bleu_score1.append(bleu.sentence_bleu([b[0].split()],y_lst1))

    y2=prediction(res[1][0])
    y2=y2.split(' ')
    y_lst2=[]
    for i in y2:
        if i=='<PAD>':
            continue
        else:
            y_lst2.append(i)
    bleu_score2.append(bleu.sentence_bleu([b[0].split()],y_lst2))

    y3=prediction(res[2][0])
    y3=y3.split(' ')
    y_lst3=[]
    for i in y3:
        if i=='<PAD>':
            continue
        else:
            y_lst3.append(i)
    bleu_score3.append(bleu.sentence_bleu([b[0].split()],y_lst3))

print("The Average Bleu Score1 is: ",sum(bleu_score1)/20)
print('>'*180)
print("The Average Bleu Score2 is: ",sum(bleu_score2)/20)
print('>'*180)
print("The Average Bleu Score3 is: ",sum(bleu_score3)/20)
print('>'*180)
```

