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, 2.33MB/s]
```

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 the data into DataFrame
```

In [4]:

```
df.head(4)#displaying top 4 four data
```

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 the last character
  x=x[:-1]
  return x
```

In [6]:

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

Out[8]:

```
(2000, 2)
```

In [9]:

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

In [10]:

```
df.shape#shape of the data
```

Out[10]:

```
(1990, 2)
```

In [151]:

```
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
print(X_train.shape)
```

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

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

Target:

In [152]:

```
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))
```

```
3034
```

In [153]:

```
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 [154]:

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

In [155]:

```
target_padded_docs_train.shape
```

Out[155]:

```
(1970, 43)
```

In [156]:

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

In [157]:

```
target_padded_docs_test.shape
```

Out[157]:

```
(20, 43)
```

Source:

In [158]:

```
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))
```

```
3702
```

In [159]:

```
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 [160]:

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

In [161]:

```
source_padded_docs_train.shape
```

Out[161]:

```
(1970, 43)
```

In [162]:

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

In [163]:

```
source_padded_docs_test.shape
```

Out[163]:

```
(20, 43)
```

In [164]:

```
#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 [165]:

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

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

In [166]:

```
#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 [167]:

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

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

Model1:

In [171]:

```
input=tf.keras.layers.Input(shape=(43,))
embed=tf.keras.layers.Embedding(source_vocab_size,512, 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
3model=tf.keras.models.Model(inputs=input,outputs=output)
model.summary()
```

```
Model: "model_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_9 (InputLayer)         [(None, 43)]              0
_________________________________________________________________
embedding_8 (Embedding)      (None, 43, 512)           1895936
_________________________________________________________________
lstm_8 (LSTM)                (None, 43, 128)           328192
_________________________________________________________________
time_distributed_16 (TimeDis (None, 43, 512)           66048
_________________________________________________________________
dropout_8 (Dropout)          (None, 43, 512)           0
_________________________________________________________________
time_distributed_17 (TimeDis (None, 43, 3035)          1556955
=================================================================
Total params: 3,847,131
Trainable params: 3,847,131
Non-trainable params: 0
_________________________________________________________________
```

In [172]:

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

In [173]:

```
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 [==============================] - 59s 57s/step - loss: 7.7430 - accuracy: 0.3208 - val_loss: 2.7519
- val_accuracy: 0.7698
Epoch 2/50
2/2 [==============================] - 1s 305ms/step - loss: 3.4479 - accuracy: 0.6728 - val_loss:
2.5025 - val_accuracy: 0.7698
Epoch 3/50
2/2 [==============================] - 1s 316ms/step - loss: 3.5213 - accuracy: 0.6728 - val_loss:
1.9935 - val_accuracy: 0.7721
Epoch 4/50
2/2 [==============================] - 1s 314ms/step - loss: 2.7795 - accuracy: 0.6730 - val_loss:
1.5858 - val_accuracy: 0.7791
Epoch 5/50
2/2 [==============================] - 1s 313ms/step - loss: 2.2776 - accuracy: 0.6807 - val_loss:
1.5165 - val_accuracy: 0.7709
Epoch 6/50
2/2 [==============================] - 1s 313ms/step - loss: 2.2367 - accuracy: 0.6729 - val_loss:
1.5061 - val_accuracy: 0.7698
Epoch 7/50
2/2 [==============================] - 1s 317ms/step - loss: 2.2036 - accuracy: 0.6739 - val_loss:
1.4808 - val_accuracy: 0.7779
Epoch 8/50
2/2 [==============================] - 1s 313ms/step - loss: 2.1772 - accuracy: 0.6805 - val_loss:
1.4521 - val_accuracy: 0.7791
Epoch 9/50
2/2 [==============================] - 1s 316ms/step - loss: 2.1343 - accuracy: 0.6825 - val_loss:
1.4368 - val_accuracy: 0.7814
Epoch 10/50
2/2 [==============================] - 1s 313ms/step - loss: 2.1044 - accuracy: 0.6819 - val_loss:
1.4336 - val_accuracy: 0.7791
```

```
Epoch 11/50
2/2 [==============================] - 1s 316ms/step - loss: 2.0708 - accuracy: 0.6837 - val_loss:
1.4310 - val_accuracy: 0.7814
Epoch 12/50
2/2 [==============================] - 1s 315ms/step - loss: 2.0483 - accuracy: 0.6863 - val_loss:
1.4018 - val_accuracy: 0.7849
Epoch 13/50
2/2 [==============================] - 1s 316ms/step - loss: 2.0173 - accuracy: 0.6879 - val_loss:
1.3817 - val_accuracy: 0.7849
Epoch 14/50
2/2 [==============================] - 1s 310ms/step - loss: 1.9866 - accuracy: 0.6885 - val_loss:
1.3591 - val_accuracy: 0.7849
Epoch 15/50
2/2 [==============================] - 1s 312ms/step - loss: 1.9527 - accuracy: 0.6894 - val_loss:
1.3418 - val_accuracy: 0.7872
Epoch 16/50
2/2 [==============================] - 1s 311ms/step - loss: 1.9202 - accuracy: 0.6916 - val_loss:
1.3293 - val_accuracy: 0.7860
Epoch 17/50
2/2 [==============================] - 1s 319ms/step - loss: 1.8916 - accuracy: 0.6921 - val_loss:
1.3174 - val_accuracy: 0.7884
Epoch 18/50
2/2 [==============================] - 1s 319ms/step - loss: 1.8620 - accuracy: 0.6944 - val_loss:
1.3016 - val_accuracy: 0.7907
Epoch 19/50
2/2 [==============================] - 1s 315ms/step - loss: 1.8346 - accuracy: 0.6972 - val_loss:
1.2932 - val_accuracy: 0.7919
Epoch 20/50
2/2 [==============================] - 1s 316ms/step - loss: 1.8075 - accuracy: 0.6993 - val_loss:
1.2764 - val_accuracy: 0.7965
Epoch 21/50
2/2 [==============================] - 1s 318ms/step - loss: 1.7780 - accuracy: 0.7020 - val_loss:
1.2680 - val_accuracy: 0.7942
Epoch 22/50
2/2 [==============================] - 1s 324ms/step - loss: 1.7495 - accuracy: 0.7045 - val_loss:
1.2495 - val_accuracy: 0.8000
Epoch 23/50
2/2 [==============================] - 1s 318ms/step - loss: 1.7181 - accuracy: 0.7080 - val_loss:
1.2368 - val_accuracy: 0.8035
Epoch 24/50
2/2 [==============================] - 1s 314ms/step - loss: 1.6881 - accuracy: 0.7115 - val_loss:
1.2230 - val_accuracy: 0.8058
Epoch 25/50
2/2 [==============================] - 1s 321ms/step - loss: 1.6586 - accuracy: 0.7140 - val_loss:
1.2126 - val_accuracy: 0.8093
Epoch 26/50
2/2 [==============================] - 1s 316ms/step - loss: 1.6247 - accuracy: 0.7168 - val_loss:
1.1898 - val_accuracy: 0.8140
Epoch 27/50
2/2 [==============================] - 1s 312ms/step - loss: 1.5962 - accuracy: 0.7213 - val_loss:
1.1732 - val_accuracy: 0.8186
Epoch 28/50
2/2 [==============================] - 1s 318ms/step - loss: 1.5594 - accuracy: 0.7254 - val_loss:
1.1697 - val_accuracy: 0.8233
Epoch 29/50
2/2 [==============================] - 1s 314ms/step - loss: 1.5303 - accuracy: 0.7302 - val_loss:
1.1395 - val_accuracy: 0.8314
Epoch 30/50
2/2 [==============================] - 1s 319ms/step - loss: 1.4953 - accuracy: 0.7371 - val_loss:
1.1235 - val_accuracy: 0.8349
Epoch 31/50
2/2 [==============================] - 1s 317ms/step - loss: 1.4593 - accuracy: 0.7407 - val_loss:
1.1209 - val_accuracy: 0.8384
Epoch 32/50
2/2 [==============================] - 1s 315ms/step - loss: 1.4236 - accuracy: 0.7477 - val_loss:
1.1059 - val_accuracy: 0.8419
Epoch 33/50
2/2 [==============================] - 1s 317ms/step - loss: 1.3867 - accuracy: 0.7514 - val_loss:
1.0881 - val_accuracy: 0.8442
Epoch 34/50
2/2 [==============================] - 1s 319ms/step - loss: 1.3504 - accuracy: 0.7583 - val_loss:
1.0822 - val_accuracy: 0.8453
Epoch 35/50
2/2 [==============================] - 1s 315ms/step - loss: 1.3140 - accuracy: 0.7633 - val_loss:
1.0658 - val_accuracy: 0.8465
Epoch 36/50
2/2 [==============================] - 1s 317ms/step - loss: 1.2742 - accuracy: 0.7691 - val_loss:
```

```
1.0565 - val_accuracy: 0.8488
Epoch 37/50
2/2 [==============================] - 1s 327ms/step - loss: 1.2396 - accuracy: 0.7743 - val_loss:
1.0437 - val_accuracy: 0.8547
Epoch 38/50
2/2 [==============================] - 1s 314ms/step - loss: 1.2227 - accuracy: 0.7780 - val_loss:
1.0457 - val_accuracy: 0.8523
Epoch 39/50
2/2 [==============================] - 1s 314ms/step - loss: 1.2551 - accuracy: 0.7719 - val_loss:
1.0567 - val_accuracy: 0.8558
Epoch 40/50
2/2 [==============================] - 1s 317ms/step - loss: 1.2022 - accuracy: 0.7853 - val_loss:
1.0322 - val_accuracy: 0.8593
Epoch 41/50
2/2 [==============================] - 1s 314ms/step - loss: 1.1494 - accuracy: 0.7873 - val_loss:
1.0276 - val_accuracy: 0.8570
Epoch 42/50
2/2 [==============================] - 1s 315ms/step - loss: 1.1155 - accuracy: 0.7881 - val_loss:
1.0399 - val_accuracy: 0.8570
Epoch 43/50
2/2 [==============================] - 1s 311ms/step - loss: 1.0756 - accuracy: 0.7968 - val_loss:
1.0274 - val_accuracy: 0.8593
Epoch 44/50
2/2 [==============================] - 1s 308ms/step - loss: 1.0426 - accuracy: 0.8013 - val_loss:
1.0311 - val_accuracy: 0.8605
Epoch 45/50
2/2 [==============================] - 1s 314ms/step - loss: 1.0029 - accuracy: 0.8069 - val_loss:
1.0222 - val_accuracy: 0.8616
Epoch 46/50
2/2 [==============================] - 1s 315ms/step - loss: 0.9696 - accuracy: 0.8117 - val_loss:
1.0364 - val_accuracy: 0.8640
Epoch 47/50
2/2 [==============================] - 1s 317ms/step - loss: 0.9377 - accuracy: 0.8166 - val_loss:
1.0423 - val_accuracy: 0.8616
Epoch 48/50
2/2 [==============================] - 1s 316ms/step - loss: 0.9083 - accuracy: 0.8210 - val_loss:
1.0496 - val_accuracy: 0.8628
Epoch 49/50
2/2 [==============================] - 1s 310ms/step - loss: 0.8767 - accuracy: 0.8248 - val_loss:
1.0535 - val_accuracy: 0.8651
Epoch 50/50
2/2 [==============================] - 1s 315ms/step - loss: 0.8457 - accuracy: 0.8306 - val_loss:
1.0778 - val_accuracy: 0.8686
```

Out[173]:

```
<tensorflow.python.keras.callbacks.History at 0x7f67c2382550>
```

In [174]:

```
#https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/
#Beam Search
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 [175]:

```
#prediction
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
#calculating bleu score using beam search where K==3
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])
  y1=y1.split(' ')
  y_lst1=[]
  for i in y1:
    if i=='<PAD>':
      continue
    else:
      y_lst1.append(i)
  print(' '.join(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)
  print(' '.join(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)
  print(' '.join(y_lst3))
  print('>'*180)
```

```
Input text:
Hey... Kick boxing on sunday is fully booked...
Actual Output:
Hey, kick boxing on Sunday is fully booked.
Predicted Output for beam==3 :
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16: RuntimeWarning: divide by zero
encountered in log
  app.launch_new_instance()
hey on sunday is booked
hey on sunday is mobile
hey on sunday is on
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Oh... I juz checked n realize my lesson is at 440...Haha, tt means i'll cya den...
Actual Output:
Oh. I just checked and realize my lesson is at 4:40. Haha, that means I'll see you then.
Predicted Output for beam==3 :
oh i just i and my lesson is at point afternoon that means i you then
oh i just i and my lesson is at point afternoon that means i see then
oh i just i and my lesson is at point haha that means i you then
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Fetch us at 7pm
Actual Output:
Fetch us at 7pm.
Predicted Output for beam==3 :
```

```
Predicted Output for beam==3 :
corner us at
million us at
drink us at
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
6pm
Actual Output:
6 pm .
Predicted Output for beam==3 :
is
at
be
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
the inauguration thing is 9am at uni cultural centre.u know abt it?
Actual Output:
The inauguration thing is 9 am. at university cultural centre. You know about it?
Predicted Output for beam==3 :
the that is at university sleepy centre you know about it
the that is at university chen centre you know about it
the that is at university cultural centre you know about it
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
dun like tat leh MR joe.
Actual Output:
Don't like that Mr. Joe.
Predicted Output for beam==3 :
don't like that i a
don't like that i i a
don't like like i a
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
How u look like
Actual Output:
How do you look like?
Predicted Output for beam==3 :
how you look like
how you go like
how are look like
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Im sorry ken!i dun chat tht things 1!
Actual Output:
I'm sorry, Ken! I don't chat that thing!
Predicted Output for beam==3 :
i'm sorry rv i don't chat things 1
i'm sorry hun i don't chat things 1
i'm sorry tina i don't chat things 1
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Joey: YOGI CARE 2 INTRO.
Actual Output:
Joey: Yogi care to introduce?
Predicted Output for beam==3 :
joey care to introduce
joey introduce to introduce
joey care to be
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Im working ... R ü ireena
Actual Output:
I'm working. Are you ireena?
Predicted Output for beam==3 :
i'm working are you
i'm working are are
i working are you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
```

```
Wat time u be home then?
Actual Output:
What time will you be home then?
Predicted Output for beam==3 :
what time you be home
what time you be home then
what time you be home i
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
No problem! Close friends call me hammy. Haha, just call me whatever you like. Good day! :)
Actual Output:
No problem! Close friends call me hammy. Haha, just call me whatever you like. Good day!
Predicted Output for beam==3 :
no another stuff friends call me haha just you me when you to good
no another stuff friends call me haha just you me when you to good is
no another stuff friends call me haha just you me help you to good
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Yun,dun nd hide k,ur classmate,chong kai qin is in same camp as me.U dare mit me at tiong bahru at
1.30pm lah
Actual Output:
Yun, don't need to hide, your classmate, Chong Kai Qin is in the same camp as me. You dare to meet me at
Tiong Bahru at 1:30pm.
Predicted Output for beam==3 :
yun don't need ok your blessed is in same camp as me you tomorrow meet me at at tiong the 1 30 shots
yun don't need ok your practice is in same camp as me you tomorrow meet me at at tiong the 1 30 shots
yun don't need ok your blessed is in same camp as me you tomorrow meet me at at tiong the 1 30 package
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey... So ür fren bro alreadi in e com eng course? Cöz i wan to find out more abt it can help me ask? Th
kz =5
Actual Output:
Hey. So your friend's brother is already in computer engineering course? Because I want to find out more
about it can help me ask? Thanks.
Predicted Output for beam==3 :
hey so your your in in the in without st my i you to to to to to to can help you go can tomorrow
hey so your your in in the without without st my i you to to to to to to can help you go can tomorrow
hey so your your in in the in without st my i you to to to to to to can help me go can tomorrow
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
It's u .. Not me ..
Actual Output:
It's you. Not me.
Predicted Output for beam==3 :
it's you not me
it's you not you
happy you not me
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Help me feed e hamsters...
Actual Output:
Help me feed the hamsters.
Predicted Output for beam==3 :
help me wrong the
help me nap the
help me feed the
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha. Oops. I didnt see your msg. Ü want to go at 9?
Actual Output:
Haha. Sorry. I didn't see your message. You want to go at 9?
Predicted Output for beam==3 :
haha i didn't see your message you you to go to four
haha i didn't see your message you you to go to four start
haha i didn't see your message you you to go to four to
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Dunno lei... Ben juz ask... He neva say... Aiyo y she like tai tai like dat always play mah jong...
Actual Output:
Don't know. Ben just asked. He never say. Why is she like an middle-aged lady like that always play
```

```
mahjong?
Predicted Output for beam==3 :
don't don't once just ask he yes say i i she i on rich owl that trip bash is
don't don't once just ask he yes say i i she it on rich owl that trip bash is
don't don't once just ask he yes say i i she i on rich owl that trip bash your
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Yup...Mit u at body shop
Actual Output:
Yes. Meet you at Body Shop.
Predicted Output for beam==3 :
yes meet you at body shop
yes meet you at body her
yes meet you at body and
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I am i am! Haha no need to save seat for rebecca
Actual Output:
I am I am! Haha, no need to save seat for Rebecca.
Predicted Output for beam==3 :
i am i am haha no need to save more for rebecca
i am i am haha no need to save more for more
i am i am haha no need to save more for on
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [176]:

```python
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 i=='<PAD>':
      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/ipykernel_launcher.py:16: RuntimeWarning: divide by zero
encountered in log
  app.launch_new_instance()
/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)
The Average Bleu Score1 is:  0.38480691187655225
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Average Bleu Score2 is:  0.36163580184089766
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Average Bleu Score3 is:  0.3734191852349319
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Model2:

In [186]:

```
input=tf.keras.layers.Input(shape=(43,))
embed=tf.keras.layers.Embedding(source_vocab_size,512, input_length=source_padded_docs_train.shape[1])(in
lstm1=tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(100, 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_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_11 (InputLayer)        [(None, 43)]              0
_________________________________________________________________
embedding_10 (Embedding)     (None, 43, 512)           1895936
_________________________________________________________________
bidirectional_1 (Bidirection (None, 43, 200)           490400
_________________________________________________________________
time_distributed_19 (TimeDis (None, 43, 3035)          610035
=================================================================
Total params: 2,996,371
Trainable params: 2,996,371
Non-trainable params: 0
_________________________________________________________________
```

In [187]:

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

In [188]:

```
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 [==============================] - 4s 874ms/step - loss: 7.6971 - accuracy: 0.3259 - val_loss:
5.2202 - val_accuracy: 0.7698
Epoch 2/50
2/2 [==============================] - 1s 286ms/step - loss: 4.6866 - accuracy: 0.6728 - val_loss:
1.9247 - val_accuracy: 0.7698
Epoch 3/50
2/2 [==============================] - 1s 296ms/step - loss: 2.7107 - accuracy: 0.6728 - val_loss:
2.0169 - val_accuracy: 0.7698
Epoch 4/50
2/2 [==============================] - 1s 291ms/step - loss: 2.9326 - accuracy: 0.6728 - val_loss:
1.9303 - val_accuracy: 0.7698
Epoch 5/50
2/2 [==============================] - 1s 291ms/step - loss: 2.7293 - accuracy: 0.6737 - val_loss:
1.8979 - val_accuracy: 0.7698
Epoch 6/50
2/2 [==============================] - 1s 289ms/step - loss: 2.5013 - accuracy: 0.6717 - val_loss:
1.6204 - val_accuracy: 0.7709
```

```
Epoch 7/50
2/2 [==============================] - 1s 286ms/step - loss: 2.2420 - accuracy: 0.6735 - val_loss:
1.4918 - val_accuracy: 0.7756
Epoch 8/50
2/2 [==============================] - 1s 285ms/step - loss: 2.1406 - accuracy: 0.6795 - val_loss:
1.4636 - val_accuracy: 0.7756
Epoch 9/50
2/2 [==============================] - 1s 290ms/step - loss: 2.1163 - accuracy: 0.6844 - val_loss:
1.4504 - val_accuracy: 0.7791
Epoch 10/50
2/2 [==============================] - 1s 286ms/step - loss: 2.1090 - accuracy: 0.6839 - val_loss:
1.4436 - val_accuracy: 0.7791
Epoch 11/50
2/2 [==============================] - 1s 284ms/step - loss: 2.0970 - accuracy: 0.6844 - val_loss:
1.4331 - val_accuracy: 0.7779
Epoch 12/50
2/2 [==============================] - 1s 287ms/step - loss: 2.0796 - accuracy: 0.6858 - val_loss:
1.4136 - val_accuracy: 0.7767
Epoch 13/50
2/2 [==============================] - 1s 293ms/step - loss: 2.0534 - accuracy: 0.6899 - val_loss:
1.3890 - val_accuracy: 0.7826
Epoch 14/50
2/2 [==============================] - 1s 285ms/step - loss: 2.0201 - accuracy: 0.6924 - val_loss:
1.3725 - val_accuracy: 0.7849
Epoch 15/50
2/2 [==============================] - 1s 284ms/step - loss: 1.9935 - accuracy: 0.6913 - val_loss:
1.3610 - val_accuracy: 0.7826
Epoch 16/50
2/2 [==============================] - 1s 288ms/step - loss: 1.9651 - accuracy: 0.6913 - val_loss:
1.3476 - val_accuracy: 0.7837
Epoch 17/50
2/2 [==============================] - 1s 286ms/step - loss: 1.9356 - accuracy: 0.6929 - val_loss:
1.3348 - val_accuracy: 0.7826
Epoch 18/50
2/2 [==============================] - 1s 289ms/step - loss: 1.9069 - accuracy: 0.6948 - val_loss:
1.3185 - val_accuracy: 0.7849
Epoch 19/50
2/2 [==============================] - 1s 289ms/step - loss: 1.8763 - accuracy: 0.6975 - val_loss:
1.2984 - val_accuracy: 0.7860
Epoch 20/50
2/2 [==============================] - 1s 290ms/step - loss: 1.8429 - accuracy: 0.6996 - val_loss:
1.2787 - val_accuracy: 0.7872
Epoch 21/50
2/2 [==============================] - 1s 290ms/step - loss: 1.8082 - accuracy: 0.7010 - val_loss:
1.2591 - val_accuracy: 0.7895
Epoch 22/50
2/2 [==============================] - 1s 290ms/step - loss: 1.7717 - accuracy: 0.7037 - val_loss:
1.2377 - val_accuracy: 0.7919
Epoch 23/50
2/2 [==============================] - 1s 285ms/step - loss: 1.7335 - accuracy: 0.7076 - val_loss:
1.2126 - val_accuracy: 0.7942
Epoch 24/50
2/2 [==============================] - 1s 286ms/step - loss: 1.6949 - accuracy: 0.7120 - val_loss:
1.1878 - val_accuracy: 0.8000
Epoch 25/50
2/2 [==============================] - 1s 289ms/step - loss: 1.6553 - accuracy: 0.7192 - val_loss:
1.1662 - val_accuracy: 0.8081
Epoch 26/50
2/2 [==============================] - 1s 291ms/step - loss: 1.6147 - accuracy: 0.7265 - val_loss:
1.1468 - val_accuracy: 0.8186
Epoch 27/50
2/2 [==============================] - 1s 288ms/step - loss: 1.5731 - accuracy: 0.7337 - val_loss:
1.1242 - val_accuracy: 0.8209
Epoch 28/50
2/2 [==============================] - 1s 291ms/step - loss: 1.5308 - accuracy: 0.7402 - val_loss:
1.1069 - val_accuracy: 0.8291
Epoch 29/50
2/2 [==============================] - 1s 289ms/step - loss: 1.4883 - accuracy: 0.7473 - val_loss:
1.0902 - val_accuracy: 0.8337
Epoch 30/50
2/2 [==============================] - 1s 292ms/step - loss: 1.4461 - accuracy: 0.7547 - val_loss:
1.0731 - val_accuracy: 0.8372
Epoch 31/50
2/2 [==============================] - 1s 288ms/step - loss: 1.4047 - accuracy: 0.7621 - val_loss:
1.0563 - val_accuracy: 0.8442
Epoch 32/50
2/2 [==============================] - 1s 284ms/step - loss: 1.3630 - accuracy: 0.7682 - val_loss:
```

```
1.0421 - val_accuracy: 0.8453
Epoch 33/50
2/2 [==============================] - 1s 298ms/step - loss: 1.3218 - accuracy: 0.7750 - val_loss:
1.0275 - val_accuracy: 0.8465
Epoch 34/50
2/2 [==============================] - 1s 288ms/step - loss: 1.2818 - accuracy: 0.7815 - val_loss:
1.0125 - val_accuracy: 0.8488
Epoch 35/50
2/2 [==============================] - 1s 286ms/step - loss: 1.2420 - accuracy: 0.7883 - val_loss:
1.0010 - val_accuracy: 0.8547
Epoch 36/50
2/2 [==============================] - 1s 290ms/step - loss: 1.2037 - accuracy: 0.7944 - val_loss:
0.9922 - val_accuracy: 0.8581
Epoch 37/50
2/2 [==============================] - 1s 290ms/step - loss: 1.1664 - accuracy: 0.8007 - val_loss:
0.9795 - val_accuracy: 0.8593
Epoch 38/50
2/2 [==============================] - 1s 288ms/step - loss: 1.1290 - accuracy: 0.8070 - val_loss:
0.9709 - val_accuracy: 0.8616
Epoch 39/50
2/2 [==============================] - 1s 289ms/step - loss: 1.0930 - accuracy: 0.8132 - val_loss:
0.9617 - val_accuracy: 0.8651
Epoch 40/50
2/2 [==============================] - 1s 297ms/step - loss: 1.0571 - accuracy: 0.8190 - val_loss:
0.9520 - val_accuracy: 0.8651
Epoch 41/50
2/2 [==============================] - 1s 290ms/step - loss: 1.0230 - accuracy: 0.8241 - val_loss:
0.9453 - val_accuracy: 0.8674
Epoch 42/50
2/2 [==============================] - 1s 289ms/step - loss: 0.9890 - accuracy: 0.8287 - val_loss:
0.9389 - val_accuracy: 0.8663
Epoch 43/50
2/2 [==============================] - 1s 287ms/step - loss: 0.9561 - accuracy: 0.8329 - val_loss:
0.9338 - val_accuracy: 0.8686
Epoch 44/50
2/2 [==============================] - 1s 291ms/step - loss: 0.9242 - accuracy: 0.8370 - val_loss:
0.9315 - val_accuracy: 0.8686
Epoch 45/50
2/2 [==============================] - 1s 284ms/step - loss: 0.8932 - accuracy: 0.8409 - val_loss:
0.9264 - val_accuracy: 0.8674
Epoch 46/50
2/2 [==============================] - 1s 285ms/step - loss: 0.8641 - accuracy: 0.8442 - val_loss:
0.9248 - val_accuracy: 0.8698
Epoch 47/50
2/2 [==============================] - 1s 293ms/step - loss: 0.8352 - accuracy: 0.8474 - val_loss:
0.9221 - val_accuracy: 0.8721
Epoch 48/50
2/2 [==============================] - 1s 302ms/step - loss: 0.8082 - accuracy: 0.8498 - val_loss:
0.9187 - val_accuracy: 0.8733
Epoch 49/50
2/2 [==============================] - 1s 287ms/step - loss: 0.7807 - accuracy: 0.8527 - val_loss:
0.9172 - val_accuracy: 0.8721
Epoch 50/50
2/2 [==============================] - 1s 287ms/step - loss: 0.7557 - accuracy: 0.8551 - val_loss:
0.9166 - val_accuracy: 0.8709
```

Out[188]:

```
<tensorflow.python.keras.callbacks.History at 0x7f67caecaa10>
```

In [189]:

```
model.fit(source_padded_docs_train,target_padded_docs_train,batch_size=1024,epochs=10,
          validation_data=(source_padded_docs_test,target_padded_docs_test))
```

```
Epoch 1/10
2/2 [==============================] - 1s 313ms/step - loss: 0.7306 - accuracy: 0.8575 - val_loss:
0.9117 - val_accuracy: 0.8733
Epoch 2/10
2/2 [==============================] - 1s 292ms/step - loss: 0.7071 - accuracy: 0.8601 - val_loss:
0.9130 - val_accuracy: 0.8744
Epoch 3/10
2/2 [==============================] - 1s 296ms/step - loss: 0.6841 - accuracy: 0.8626 - val_loss:
0.9143 - val_accuracy: 0.8756
Epoch 4/10
2/2 [==============================] - 1s 286ms/step - loss: 0.6617 - accuracy: 0.8648 - val_loss:
0.9150 - val_accuracy: 0.8733
Epoch 5/10
2/2 [==============================] - 1s 290ms/step - loss: 0.6407 - accuracy: 0.8671 - val_loss:
0.9149 - val_accuracy: 0.8733
Epoch 6/10
2/2 [==============================] - 1s 294ms/step - loss: 0.6204 - accuracy: 0.8699 - val_loss:
0.9132 - val_accuracy: 0.8756
Epoch 7/10
2/2 [==============================] - 1s 293ms/step - loss: 0.6007 - accuracy: 0.8718 - val_loss:
0.9126 - val_accuracy: 0.8767
Epoch 8/10
2/2 [==============================] - 1s 289ms/step - loss: 0.5823 - accuracy: 0.8743 - val_loss:
0.9142 - val_accuracy: 0.8756
Epoch 9/10
2/2 [==============================] - 1s 288ms/step - loss: 0.5641 - accuracy: 0.8775 - val_loss:
0.9143 - val_accuracy: 0.8744
Epoch 10/10
2/2 [==============================] - 1s 294ms/step - loss: 0.5471 - accuracy: 0.8793 - val_loss:
0.9148 - val_accuracy: 0.8744
```

Out[189]:

```
<tensorflow.python.keras.callbacks.History at 0x7f67caf1de90>
```

In [190]:

```
#https://machinelearningmastery.com/beam-search-decoder-natural-language-processing/
#Beam Search

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 [191]:

```
#prediction
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
#calculating bleu score using beam search where K==3
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])
  y1=y1.split(' ')
  y_lst1=[]
  for i in y1:
    if i=='<PAD>':
      continue
    else:
      y_lst1.append(i)
  print(' '.join(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)
  print(' '.join(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)
  print(' '.join(y_lst3))
  print('>'*180)
```

```
Input text:
Hey... Kick boxing on sunday is fully booked...
Actual Output:
Hey, kick boxing on Sunday is fully booked.
Predicted Output for beam==3 :
hey on sunday is jo
hey on sunday is green
hey on sunday is 2nd
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Oh... I juz checked n realize my lesson is at 440...Haha, tt means i'll cya den...
Actual Output:
Oh. I just checked and realize my lesson is at 4:40. Haha, that means I'll see you then.
Predicted Output for beam==3 :
oh i just i and my lesson is at 4 haha then i'll i'll you then
oh i just i and my lesson is at 4 haha then i'll i'll you you
oh i just i and my lesson is at 4 haha then i'll i'll you meet
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Fetch us at 7pm
Actual Output:
Fetch us at 7pm.
Predicted Output for beam==3 :
fetch us at tiring
fetch us at 40
fetch us at minutes
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
6pm
Actual Output:
6 pm .
Predicted Output for beam==3 :
after
taxi
esplanade
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
the inauguration thing is 9am at uni cultural centre.u know abt it?
Actual Output:
The inauguration thing is 9 am. at university cultural centre. You know about it?
Predicted Output for beam==3 :
the thing is at university cultural centre you you about it
the thing is at university cultural centre you about it
the thing is at university group centre you you about it
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
dun like tat leh MR joe.
Actual Output:
Don't like that Mr. Joe.
Predicted Output for beam==3 :
don't like that like i hitting
don't like like like i hitting
don't like that like i letter
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
How u look like
Actual Output:
How do you look like?
Predicted Output for beam==3 :
how you you like
how you you
how do you like
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Im sorry ken!i dun chat tht things 1!
Actual Output:
I'm sorry, Ken! I don't chat that thing!
Predicted Output for beam==3 :
i'm sorry buys i don't chat things 1
i'm sorry devin i don't chat things 1
i'm sorry harry i don't chat things 1
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Joey: YOGI CARE 2 INTRO.
Actual Output:
Joey: Yogi care to introduce?
Predicted Output for beam==3 :
joey care to introduce
joey care 2 introduce
joey care already introduce
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Im working ... R ü ireena
Actual Output:
I'm working. Are you ireena?
Predicted Output for beam==3 :
i'm working are you
i'm working are
i'm working you you
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Wat time u be home then?
Actual Output:
What time will you be home then?
Predicted Output for beam==3 :
what time you be home then
what time you you home then
what time you be home early
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
No problem! Close friends call me hammy. Haha, just call me whatever you like. Good day! :)
Actual Output:
No problem! Close friends call me hammy. Haha, just call me whatever you like. Good day!
Predicted Output for beam==3 :
no no close friends call me haha i call me whatever you me good day
```

```
no no close friends call me haha i call me whatever you good day
no no close friends call me haha i call me whatever you i good day
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Yun,dun nd hide k,ur classmate,chong kai qin is in same camp as me.U dare mit me at tiong bahru at
1.30pm lah
Actual Output:
Yun, don't need to hide, your classmate, Chong Kai Qin is in the same camp as me. You dare to meet me at
Tiong Bahru at 1:30pm.
Predicted Output for beam==3 :
yun don't need i'm your seldom is of same camp as me you you meet me at at tiong at at 30 at
yun don't need i'm your seldom is in same camp as me you you meet me at at tiong at at 30 at
yun don't need i'm your cover is of same camp as me you you meet me at at tiong at at 30 at
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Hey... So ür fren bro alreadi in e com eng course? Cöz i wan to find out more abt it can help me ask? Th
kz =5
Actual Output:
Hey. So your friend's brother is already in computer engineering course? Because I want to find out more
about it can help me ask? Thanks.
Predicted Output for beam==3 :
hey so your the my in the moon my course i i want to to send to about be can can me ask thanks
hey so your the my in the moon my course i i want to to to to about be can can me ask thanks
hey so your the my in the moon my course i i want to to send to about be can can me ask
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
It's u .. Not me ..
Actual Output:
It's you. Not me.
Predicted Output for beam==3 :
it's you not me
it's you you me
it's you are me
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Help me feed e hamsters...
Actual Output:
Help me feed the hamsters.
Predicted Output for beam==3 :
help me lecture the
help me lecture
help me nopez the
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Haha. Oops. I didnt see your msg. Ü want to go at 9?
Actual Output:
Haha. Sorry. I didn't see your message. You want to go at 9?
Predicted Output for beam==3 :
haha i didn't see your message you you to go at 9
haha i didn't see your message you want to go at 9
haha i didn't see your message you you to to at 9
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Dunno lei... Ben juz ask... He neva say... Aiyo y she like tai tai like dat always play mah jong...
Actual Output:
Don't know. Ben just asked. He never say. Why is she like an middle-aged lady like that always play
mahjong?
Predicted Output for beam==3 :
don't don't ben just ask he never says she why she always always always like that always play how
don't know ben just ask he never says she why she always always always like that always play how
don't don't ben just ask he never says she why she always always always like that always play
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
Yup...Mit u at body shop
Actual Output:
Yes. Meet you at Body Shop.
Predicted Output for beam==3 :
yes meet you at she's
yes meet you at wear
yes meet you at she's shop
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Input text:
I am i am! Haha no need to save seat for rebecca
Actual Output:
I am I am! Haha, no need to save seat for Rebecca.
Predicted Output for beam==3 :
i am i am haha no need to save seat for rebecca to
i am i am haha no need to save seat for rebecca i
i am i am don't no need to save seat for rebecca to
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [192]:

```
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 i=='<PAD>':
      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 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)
The Average Bleu Score1 is:  0.3672876018410912
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Average Bleu Score2 is:  0.336078085901221
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
The Average Bleu Score3 is:  0.3697109749269815
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [ ]: