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, 2.50MB/s]
```

In [2]:
```
!pip install kaggle
```
```
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from kaggle) (4.41.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.23.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.24.3)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages (from kaggle) (1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from kaggle) (2021.5.30)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages (from kaggle) (2.8.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages (from kaggle) (5.0.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-packages (from python-slugify
->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle
) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->kaggle) (2.
10)
```

In [3]:
```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 /root/.kaggle/kaggle.json
!kaggle datasets download -d yekenot/fasttext-crawl-300d-2m
```
```
Downloading fasttext-crawl-300d-2m.zip to /content
 99% 1.43G/1.44G [00:16<00:00, 75.8MB/s]
100% 1.44G/1.44G [00:16<00:00, 91.7MB/s]
```

In [4]:
```
!7z e fasttext-crawl-300d-2m.zip -o/content -r
```
```
7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Xeon(R) CPU @ 2.00GHz (5065
3),ASM,AES-NI)

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

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

  0%      0% - crawl-300d-2M.vec                          1% - crawl-300d-2M.vec                          2% - cr
awl-300d-2M.vec                          3% - crawl-300d-2M.vec                          4% - crawl-300d-2M.vec
  5% - crawl-300d-2M.vec                          6% - crawl-300d-2M.vec                          7% - crawl-300d
-2M.vec                          8% - crawl-300d-2M.vec                          9% - crawl-300d-2M.vec
 10% - crawl-300d-2M.vec                         11% - crawl-300d-2M.vec                         12% - crawl-300d
-2M.vec                         13% - crawl-300d-2M.vec                         14% - crawl-300d-2M.vec
 15% - crawl-300d-2M.vec                         16% - crawl-300d-2M.vec                         17% - crawl-300d
-2M.vec                         18% - crawl-300d-2M.vec                         19% - crawl-300d-2M.vec
 20% - crawl-300d-2M.vec                         21% - crawl-300d-2M.vec                         22% - crawl-300d
-2M.vec                         23% - crawl-300d-2M.vec                         24% - crawl-300d-2M.vec
 25% - crawl-300d-2M.vec                         26% - crawl-300d-2M.vec                         27% - crawl-300d
-2M.vec                         28% - crawl-300d-2M.vec                         29% - crawl-300d-2M.vec
 30% - crawl-300d-2M.vec                         31% - crawl-300d-2M.vec                         32% - crawl-300d
-2M.vec                         33% - crawl-300d-2M.vec                         34% - crawl-300d-2M.vec
 35% - crawl-300d-2M.vec                         36% - crawl-300d-2M.vec                         37% - crawl-300d
-2M.vec                         38% - crawl-300d-2M.vec                         39% - crawl-300d-2M.vec
 40% - crawl-300d-2M.vec                         41% - crawl-300d-2M.vec                         42% - crawl-300d
-2M.vec                         43% - crawl-300d-2M.vec                         44% - crawl-300d-2M.vec
 45% - crawl-300d-2M.vec                         46% - crawl-300d-2M.vec                         47% - crawl-300d
-2M.vec                         48% - crawl-300d-2M.vec                         49% - crawl-300d-2M.vec
 50% - crawl-300d-2M.vec                         51% - crawl-300d-2M.vec                         52% - crawl-300d
-2M.vec                         53% - crawl-300d-2M.vec                         54% - crawl-300d-2M.vec
 55% - crawl-300d-2M.vec                         56% - crawl-300d-2M.vec                         57% - crawl-300d
-2M.vec                         58% - crawl-300d-2M.vec                         59% - crawl-300d-2M.vec
 60% - crawl-300d-2M.vec                         61% - crawl-300d-2M.vec                         62% - crawl-300d
-2M.vec                         63% - crawl-300d-2M.vec                         64% - crawl-300d-2M.vec
 65% - crawl-300d-2M.vec                         66% - crawl-300d-2M.vec                         67% - crawl-300d
-2M.vec                         68% - crawl-300d-2M.vec                         69% - crawl-300d-2M.vec
 70% - crawl-300d-2M.vec                         71% - crawl-300d-2M.vec                         72% - crawl-300d
```

```
-2M.vec                         73% - crawl-300d-2M.vec                         74% - crawl-300d-2M.vec
 75% - crawl-300d-2M.vec                         76% - crawl-300d-2M.vec                         77% - crawl-300d
-2M.vec                         78% - crawl-300d-2M.vec                         79% - crawl-300d-2M.vec
 80% - crawl-300d-2M.vec                         81% - crawl-300d-2M.vec                         82% - crawl-300d
-2M.vec                         83% - crawl-300d-2M.vec                         84% - crawl-300d-2M.vec
 85% - crawl-300d-2M.vec                         86% - crawl-300d-2M.vec                         87% - crawl-300d
-2M.vec                         88% - crawl-300d-2M.vec                         89% - crawl-300d-2M.vec
 90% - crawl-300d-2M.vec                         91% - crawl-300d-2M.vec                         92% - crawl-300d
-2M.vec                         93% - crawl-300d-2M.vec                         94% - crawl-300d-2M.vec
 95% - crawl-300d-2M.vec                         96% - crawl-300d-2M.vec                         97% - crawl-300d
-2M.vec                         98% - crawl-300d-2M.vec                         99% - crawl-300d-2M.vec
100% - crawl-300d-2M.vec                        Everything is Ok

Size:       4516698366
Compressed: 1545551987
```

In [5]:

```python
#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 [6]:

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

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

In [7]:

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

In [8]:

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

Out[8]:

	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 [9]:

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

In [10]:

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

In [11]:

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

Out[11]:

	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 [12]:

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

Out[12]:

```
(2000, 2)
```

```
Out[12]: (2000, 2)
```

In [13]:

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

In [14]:

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

Out[14]:

```
(1990, 2)
```

In [15]:

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

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

Target:

In [16]:

```
target_tokenizer= Tokenizer(lower=False)#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))
```

```
3600
```

In [17]:

```
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 [18]:

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

In [19]:

```
target_padded_docs_train.shape
```

Out[19]:

```
(1970, 43)
```

In [20]:

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

In [21]:

```
target_padded_docs_test.shape
```

Out[21]:

```
(20, 43)
```

Source:

In [22]:

```
source_tokenizer= Tokenizer(lower=False)#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))
```

```
4623
```

In [23]:

```
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 [24]:

```
source_padded_docs_train = pad_sequences(source_encoded_docs_train,maxlen=target_padded_docs_train.shape[1],paddi
```

In [25]:

```
source_padded_docs_train.shape
```

Out[25]:

```
(1970, 43)
```

In [26]:

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

In [27]:

```
source_padded_docs_test.shape
```

Out[27]:

```
(20, 43)
```

In [28]:

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

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 [29]:

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

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

In [30]:

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

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 [31]:

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

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

In [32]:

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

Model:

In [33]:

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

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 43)]              0         
_________________________________________________________________
embedding (Embedding)        (None, 43, 300)           1387200   
_________________________________________________________________
bidirectional (Bidirectional (None, 43, 256)           439296    
_________________________________________________________________
time_distributed (TimeDistri (None, 43, 3601)          925457    
=================================================================
Total params: 2,751,953
Trainable params: 1,364,753
Non-trainable params: 1,387,200
_________________________________________________________________
```

In [34]:

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

In [35]:

```
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 [==============================] - 10s 907ms/step - loss: 8.0933 - accuracy: 0.3287 - val_loss: 6.4809 - val_
accuracy: 0.7047
Epoch 2/50
2/2 [==============================] - 1s 260ms/step - loss: 5.7948 - accuracy: 0.6822 - val_loss: 3.0150 - val_a
ccuracy: 0.6977
```

```
Epoch 3/50
2/2 [==============================] - 1s 265ms/step - loss: 3.2016 - accuracy: 0.6735 - val_loss: 2.9736 - val_a
ccuracy: 0.6977
Epoch 4/50
2/2 [==============================] - 1s 261ms/step - loss: 3.2845 - accuracy: 0.6735 - val_loss: 2.5922 - val_a
ccuracy: 0.6977
Epoch 5/50
2/2 [==============================] - 1s 263ms/step - loss: 2.8372 - accuracy: 0.6745 - val_loss: 2.3567 - val_a
ccuracy: 0.7012
Epoch 6/50
2/2 [==============================] - 1s 264ms/step - loss: 2.6457 - accuracy: 0.6775 - val_loss: 2.3243 - val_a
ccuracy: 0.7012
Epoch 7/50
2/2 [==============================] - 1s 266ms/step - loss: 2.5460 - accuracy: 0.6775 - val_loss: 2.1505 - val_a
ccuracy: 0.7012
Epoch 8/50
2/2 [==============================] - 1s 268ms/step - loss: 2.3997 - accuracy: 0.6771 - val_loss: 2.0940 - val_a
ccuracy: 0.7012
Epoch 9/50
2/2 [==============================] - 1s 265ms/step - loss: 2.3458 - accuracy: 0.6771 - val_loss: 2.0330 - val_a
ccuracy: 0.7012
Epoch 10/50
2/2 [==============================] - 1s 265ms/step - loss: 2.2635 - accuracy: 0.6778 - val_loss: 1.9568 - val_a
ccuracy: 0.7058
Epoch 11/50
2/2 [==============================] - 1s 264ms/step - loss: 2.1752 - accuracy: 0.6854 - val_loss: 1.9030 - val_a
ccuracy: 0.7105
Epoch 12/50
2/2 [==============================] - 1s 260ms/step - loss: 2.1178 - accuracy: 0.6864 - val_loss: 1.8722 - val_a
ccuracy: 0.7128
Epoch 13/50
2/2 [==============================] - 1s 266ms/step - loss: 2.0808 - accuracy: 0.6887 - val_loss: 1.8578 - val_a
ccuracy: 0.7151
Epoch 14/50
2/2 [==============================] - 1s 267ms/step - loss: 2.0554 - accuracy: 0.6938 - val_loss: 1.8461 - val_a
ccuracy: 0.7151
Epoch 15/50
2/2 [==============================] - 1s 267ms/step - loss: 2.0285 - accuracy: 0.6963 - val_loss: 1.8355 - val_a
ccuracy: 0.7151
Epoch 16/50
2/2 [==============================] - 1s 270ms/step - loss: 2.0027 - accuracy: 0.6962 - val_loss: 1.8244 - val_a
ccuracy: 0.7163
Epoch 17/50
2/2 [==============================] - 1s 266ms/step - loss: 1.9774 - accuracy: 0.6987 - val_loss: 1.8055 - val_a
ccuracy: 0.7186
Epoch 18/50
2/2 [==============================] - 1s 265ms/step - loss: 1.9501 - accuracy: 0.7008 - val_loss: 1.7804 - val_a
ccuracy: 0.7209
Epoch 19/50
2/2 [==============================] - 1s 269ms/step - loss: 1.9222 - accuracy: 0.7027 - val_loss: 1.7632 - val_a
ccuracy: 0.7221
Epoch 20/50
2/2 [==============================] - 1s 270ms/step - loss: 1.8913 - accuracy: 0.7047 - val_loss: 1.7485 - val_a
ccuracy: 0.7221
Epoch 21/50
2/2 [==============================] - 1s 271ms/step - loss: 1.8590 - accuracy: 0.7072 - val_loss: 1.7303 - val_a
ccuracy: 0.7244
Epoch 22/50
2/2 [==============================] - 1s 268ms/step - loss: 1.8256 - accuracy: 0.7099 - val_loss: 1.7073 - val_a
ccuracy: 0.7267
Epoch 23/50
2/2 [==============================] - 1s 266ms/step - loss: 1.7898 - accuracy: 0.7131 - val_loss: 1.6811 - val_a
ccuracy: 0.7267
Epoch 24/50
2/2 [==============================] - 1s 269ms/step - loss: 1.7533 - accuracy: 0.7158 - val_loss: 1.6522 - val_a
ccuracy: 0.7291
Epoch 25/50
2/2 [==============================] - 1s 267ms/step - loss: 1.7155 - accuracy: 0.7190 - val_loss: 1.6274 - val_a
ccuracy: 0.7302
Epoch 26/50
2/2 [==============================] - 1s 272ms/step - loss: 1.6773 - accuracy: 0.7230 - val_loss: 1.6027 - val_a
ccuracy: 0.7372
Epoch 27/50
2/2 [==============================] - 1s 265ms/step - loss: 1.6390 - accuracy: 0.7269 - val_loss: 1.5762 - val_a
ccuracy: 0.7442
Epoch 28/50
2/2 [==============================] - 1s 264ms/step - loss: 1.5987 - accuracy: 0.7323 - val_loss: 1.5542 - val_a
ccuracy: 0.7465
Epoch 29/50
2/2 [==============================] - 1s 266ms/step - loss: 1.5592 - accuracy: 0.7374 - val_loss: 1.5279 - val_a
ccuracy: 0.7535
Epoch 30/50
2/2 [==============================] - 1s 267ms/step - loss: 1.5180 - accuracy: 0.7424 - val_loss: 1.5043 - val_a
```

```
ccuracy: 0.7570
Epoch 31/50
2/2 [==============================] - 1s 265ms/step - loss: 1.4774 - accuracy: 0.7475 - val_loss: 1.4805 - val_a
ccuracy: 0.7674
Epoch 32/50
2/2 [==============================] - 1s 265ms/step - loss: 1.4354 - accuracy: 0.7532 - val_loss: 1.4580 - val_a
ccuracy: 0.7686
Epoch 33/50
2/2 [==============================] - 1s 273ms/step - loss: 1.3944 - accuracy: 0.7580 - val_loss: 1.4315 - val_a
ccuracy: 0.7756
Epoch 34/50
2/2 [==============================] - 1s 273ms/step - loss: 1.3539 - accuracy: 0.7635 - val_loss: 1.4014 - val_a
ccuracy: 0.7802
Epoch 35/50
2/2 [==============================] - 1s 266ms/step - loss: 1.3140 - accuracy: 0.7694 - val_loss: 1.3913 - val_a
ccuracy: 0.7872
Epoch 36/50
2/2 [==============================] - 1s 266ms/step - loss: 1.2735 - accuracy: 0.7754 - val_loss: 1.3786 - val_a
ccuracy: 0.7895
Epoch 37/50
2/2 [==============================] - 1s 274ms/step - loss: 1.2362 - accuracy: 0.7808 - val_loss: 1.3680 - val_a
ccuracy: 0.7919
Epoch 38/50
2/2 [==============================] - 1s 268ms/step - loss: 1.1998 - accuracy: 0.7854 - val_loss: 1.3391 - val_a
ccuracy: 0.7930
Epoch 39/50
2/2 [==============================] - 1s 273ms/step - loss: 1.1620 - accuracy: 0.7915 - val_loss: 1.3353 - val_a
ccuracy: 0.7965
Epoch 40/50
2/2 [==============================] - 1s 272ms/step - loss: 1.1271 - accuracy: 0.7971 - val_loss: 1.3174 - val_a
ccuracy: 0.8012
Epoch 41/50
2/2 [==============================] - 1s 269ms/step - loss: 1.0928 - accuracy: 0.8023 - val_loss: 1.3064 - val_a
ccuracy: 0.8035
Epoch 42/50
2/2 [==============================] - 1s 273ms/step - loss: 1.0604 - accuracy: 0.8067 - val_loss: 1.2964 - val_a
ccuracy: 0.8070
Epoch 43/50
2/2 [==============================] - 1s 269ms/step - loss: 1.0307 - accuracy: 0.8116 - val_loss: 1.2836 - val_a
ccuracy: 0.8047
Epoch 44/50
2/2 [==============================] - 1s 269ms/step - loss: 1.0010 - accuracy: 0.8157 - val_loss: 1.2868 - val_a
ccuracy: 0.8093
Epoch 45/50
2/2 [==============================] - 1s 270ms/step - loss: 0.9764 - accuracy: 0.8199 - val_loss: 1.2809 - val_a
ccuracy: 0.8058
Epoch 46/50
2/2 [==============================] - 1s 266ms/step - loss: 0.9500 - accuracy: 0.8226 - val_loss: 1.2702 - val_a
ccuracy: 0.8035
Epoch 47/50
2/2 [==============================] - 1s 268ms/step - loss: 0.9238 - accuracy: 0.8262 - val_loss: 1.2636 - val_a
ccuracy: 0.8023
Epoch 48/50
2/2 [==============================] - 1s 267ms/step - loss: 0.9016 - accuracy: 0.8287 - val_loss: 1.2607 - val_a
ccuracy: 0.8035
Epoch 49/50
2/2 [==============================] - 1s 271ms/step - loss: 0.8787 - accuracy: 0.8306 - val_loss: 1.2577 - val_a
ccuracy: 0.8058
Epoch 50/50
2/2 [==============================] - 1s 272ms/step - loss: 0.8584 - accuracy: 0.8339 - val_loss: 1.2599 - val_a
ccuracy: 0.8058
```

Out[35]:

```
<tensorflow.python.keras.callbacks.History at 0x7f002465ccd0>
```

In [36]:

```
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 290ms/step - loss: 0.8384 - accuracy: 0.8356 - val_loss: 1.2536 - val_a
ccuracy: 0.8047
Epoch 2/10
2/2 [==============================] - 1s 266ms/step - loss: 0.8204 - accuracy: 0.8374 - val_loss: 1.2511 - val_a
ccuracy: 0.8023
Epoch 3/10
2/2 [==============================] - 1s 267ms/step - loss: 0.8040 - accuracy: 0.8393 - val_loss: 1.2533 - val_a
ccuracy: 0.8058
Epoch 4/10
2/2 [==============================] - 1s 272ms/step - loss: 0.7863 - accuracy: 0.8412 - val_loss: 1.2699 - val_a
ccuracy: 0.8047
Epoch 5/10
```

```
2/2 [==============================] - 1s 272ms/step - loss: 0.7721 - accuracy: 0.8430 - val_loss: 1.2544 - val_a
ccuracy: 0.8058
Epoch 6/10
2/2 [==============================] - 1s 269ms/step - loss: 0.7573 - accuracy: 0.8446 - val_loss: 1.2635 - val_a
ccuracy: 0.8023
Epoch 7/10
2/2 [==============================] - 1s 267ms/step - loss: 0.7431 - accuracy: 0.8470 - val_loss: 1.2635 - val_a
ccuracy: 0.8047
Epoch 8/10
2/2 [==============================] - 1s 271ms/step - loss: 0.7285 - accuracy: 0.8484 - val_loss: 1.2744 - val_a
ccuracy: 0.8035
Epoch 9/10
2/2 [==============================] - 1s 272ms/step - loss: 0.7168 - accuracy: 0.8497 - val_loss: 1.2709 - val_a
ccuracy: 0.8047
Epoch 10/10
2/2 [==============================] - 1s 269ms/step - loss: 0.7004 - accuracy: 0.8513 - val_loss: 1.2611 - val_a
ccuracy: 0.8023
```

Out[36]:

```
<tensorflow.python.keras.callbacks.History at 0x7f0023c68c50>
```

In [50]:

```python
from datetime import datetime
def prediction(x):

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

  y=' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
  return y
def function1(x):
  start = datetime.now()
  encoded=source_tokenizer.texts_to_sequences(x)
  padded=pad_sequences(encoded,maxlen=43,padding="post")
  padded=padded.reshape((*padded.shape,1))
  x=model.predict(padded)
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  print("The time for evaluation:"+str(datetime.now() - start))
  return ' '.join(y_lst)
```

In [51]:

```python
x='wht r u doin?'
function1([x])
```

```
The time for evaluation:0:00:00.040164
```

Out[51]:

```
'What are you doing'
```

In [52]:

```python
def prediction(x):

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

  y=' '.join([index_to_words[prediction] for prediction in np.argmax(x, 1)])
  return y
def function2(x,y_true):
  start = datetime.now()
  encoded=source_tokenizer.texts_to_sequences(x)
  padded=pad_sequences(encoded,maxlen=43,padding="post")
  padded=padded.reshape((*padded.shape,1))
  x=model.predict(padded)
  y=prediction(x[0])
  y=y.split(' ')
  y_lst=[]
  for i in y:
    if i=='<PAD>':
      continue
    else:
      y_lst.append(i)
  bleu_score=bleu.sentence_bleu([y_true.split(),],y_lst)
  print("The time for evaluation:"+str(datetime.now() - start))
  return bleu_score
```

In [53]:

```python
x='wht r u doin?'
y='What are you doing?'
```

In [54]:

```
import nltk.translate.bleu_score as bleu
function2([x],y)
```

```
The time for evaluation:0:00:00.040790
/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)
```

Out[54]:

```
0.7071067811865476
```

In [42]:

```
#https://github.com/bhattbhavesh91/tflite-tutorials/blob/master/tflite-part-2.ipynb
def get_file_size(file_path):
    size = os.path.getsize(file_path)
    return size
```

In [43]:

```
def convert_bytes(size, unit=None):
    if unit == "KB":
        return print('File size: ' + str(round(size / 1024, 3)) + ' Kilobytes')
    elif unit == "MB":
        return print('File size: ' + str(round(size / (1024 * 1024), 3)) + ' Megabytes')
    else:
        return print('File size: ' + str(size) + ' bytes')
```

Before Quantization:

In [44]:

```
import os
model_name="final_model.h5"
model.save(model_name)
convert_bytes(get_file_size(model_name),"MB")
```

```
File size: 20.959 Megabytes
```

After Quantization:

In [45]:

```
TF_LITE_MODEL_FILE_NAME = "tf_lite_model.tflite"

tf_lite_converter = tf.lite.TFLiteConverter.from_keras_model(model)
tf_lite_converter.optimizations = [tf.lite.Optimize.OPTIMIZE_FOR_SIZE]
# tf_lite_converter.optimizations = [tf.lite.Optimize.DEFAULT]
# tf_lite_converter.target_spec.supported_types = [tf.float16]
tflite_model = tf_lite_converter.convert()
```

```
WARNING:absl:Found untraced functions such as lstm_cell_1_layer_call_fn, lstm_cell_1_layer_call_and_return_condit
ional_losses, lstm_cell_2_layer_call_fn, lstm_cell_2_layer_call_and_return_conditional_losses, lstm_cell_1_layer_
call_fn while saving (showing 5 of 10). These functions will not be directly callable after loading.
INFO:tensorflow:Assets written to: /tmp/tmpqwp9h5wy/assets
INFO:tensorflow:Assets written to: /tmp/tmpqwp9h5wy/assets
```

In [46]:

```
tflite_model_name = TF_LITE_MODEL_FILE_NAME
open(tflite_model_name, "wb").write(tflite_model)
```

Out[46]:

```
2811504
```

In [47]:

```
convert_bytes(get_file_size(TF_LITE_MODEL_FILE_NAME), "MB")
```

```
File size: 2.681 Megabytes
```

Streamlit DemoVideo Link:-https://youtu.be/Tm5MaWqa7OA

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js