Sequence to sequence implementation

In [2]:

```
!pip install contractions
```

```
Collecting contractions
  Downloading https://files.pythonhosted.org/packages/0a/04/d5e0bb9f2cef5d
15616ebf68087a725c5dbdd71bd422bcfb35d709f98ce7/contractions-0.0.48-py2.py3
-none-any.whl
Collecting textsearch>=0.0.21
  Downloading https://files.pythonhosted.org/packages/d3/fe/021d7d76961b5c
eb9f8d022c4138461d83beff36c3938dc424586085e559/textsearch-0.0.21-py2.py3-n
one-any.whl
Collecting anyascii
  Downloading https://files.pythonhosted.org/packages/09/c7/61370d9e3c3494
78e89a5554c1e5d9658e1e3116cc4f2528f568909ebdf1/anyascii-0.1.7-py3-none-an
y.whl (260kB)
     |████████████████████████████████| 266kB 4.3MB/s
Collecting pyahocorasick
  Downloading https://files.pythonhosted.org/packages/4a/92/b3c70b8cf2b76f
7e3e8b7243d6f06f7cb3bab6ada237b1bce57604c5c519/pyahocorasick-1.4.1.tar.gz
(321kB)
     |████████████████████████████████| 327kB 6.1MB/s
Building wheels for collected packages: pyahocorasick
  Building wheel for pyahocorasick (setup.py) ... done
  Created wheel for pyahocorasick: filename=pyahocorasick-1.4.1-cp36-cp36m
-linux_x86_64.whl size=84341 sha256=451388d2cc4a028631e4bea9d2a3b86a8dc4cc
c2737c3080cf0ae11bc93eb054
  Stored in directory: /root/.cache/pip/wheels/e4/ab/f7/cb39270df8f6126f3d
d4c33d302357167086db460968cfc80c
Successfully built pyahocorasick
Installing collected packages: anyascii, pyahocorasick, textsearch, contra
ctions
Successfully installed anyascii-0.1.7 contractions-0.0.48 pyahocorasick-1.
4.1 textsearch-0.0.21
```

There will be some functions that start with the word "grader" ex: grader_check_encoder(), grader_check_attention(), grader_onestepdecoder() etc, you should not change those function definition.

Every Grader function has to return True.

Note 1: There are many blogs on the attention mechanisum which might be misleading you, so do read the references completly and after that only please check the internet. The best things is to read the research papers and try to implement it on your own.

Note 2: To complete this assignment, the reference that are mentioned will be enough.

Note 3: If you are starting this assignment, you might have completed minimum of 20 assignment. If you are still not able to implement this algorithm you might have rushed in the previous assignments with out learning much and didn't spend your time productively.

Task -1: Simple Encoder and Decoder

Implement simple Encoder-Decoder model

1. Download the **Italian** to **English** translation dataset from here (http://www.manythings.org/anki/ita-eng.zip)
2. You will find **ita.txt** file in that ZIP, you can read that data using python and preprocess that data this way only:

```
Encoder input: "<start> vado a scuola <end>"
Decoder input: "<start> i am going school"
Decoder output: "i am going school <end>"
```

3. You have to implement a simple Encoder and Decoder architecture
4. Use BLEU score as metric to evaluate your model. You can use any loss function you need.
5. You have to use Tensorboard to plot the Graph, Scores and histograms of gradients.
6. a. Check the reference notebook
   b. Resource 2 (https://medium.com/analytics-vidhya/understand-sequence-to-sequence-models-in-a-more-intuitive-way-1d517d8795bb)

Load the data

In [3]:

```
import tensorflow as tf
import warnings
import pandas as pd
from tqdm import tqdm
import contractions
import re
import numpy as np
```

In [4]:

```
gpus = tf.config.experimental.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(gpus[0], True)
warnings.filterwarnings('ignore')
data = open('ita.txt', 'r', encoding = 'utf-8').read().split('\n')
print(len(data))
```

```
343814
```

In [5]:

```
data_f = []
for i in range(100000):
    try:
        english, italin = data[i].split('\t')[0:2]
        data_f.append([english, italin])
    except:
        pass
data_f = pd.DataFrame(data_f, columns = ['english', 'italin'])
data_f.head()
```

Out[5]:

	english	italin
0	Hi.	Ciao!
1	Run!	Corri!
2	Run!	Corra!
3	Run!	Correte!
4	Who?	Chi?

Preprocess data

In [6]:

```
"""Ref: https://www.geeksforgeeks.org/nlp-expand-contractions-in-txt-processing/"""
def preprocess(txt):
    txt = txt.lower()
    for a, b in enumerate(txt.split()):
        try:
            if len(re.findall('[^\w\d\ "]', b))> 0:
                txt = re.sub(b, contractions.fix(b), txt)
            txt = re.sub('[^A-Za-z0-9èìò ]+', '', txt)
        except:
            return np.nan
    return txt

dat_eng = data_f.english.astype('str').apply(lambda x: preprocess(x))
dat_ita = data_f.italin.astype('str').apply(lambda x: preprocess(x))
data_preprocessing = pd.DataFrame(data = np.array([dat_eng, dat_ita]).T, columns = ['en
glish', 'italin'])
```

Implement custom encoder decoder

Encoder

In [7]:

```
from tensorflow.keras.layers import Embedding, LSTM, Dense
import tensorflow as tf
class Encoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns encoder-outputs,encoder_fi
nal_state_h,encoder_final_state_c
    '''
    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        self.lstm_size = lstm_size
        #Initialize Embedding layer
        self.enc_embed = Embedding(input_dim = inp_vocab_size, output_dim = embedding_s
ize, input_length= input_length)
        #Intialize Encoder LSTM layer
        self.enc_lstm = LSTM(lstm_size, return_sequences = True, return_state = True)

    def call(self,input_sequence,states):
        embedding = self.enc_embed(input_sequence)
        output_state, enc_h, enc_c = self.enc_lstm(embedding, initial_state = states)
        return output_state, enc_h, enc_c

    def initialize_states(self,batch_size):
        return [tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size, self.lstm
_size))]
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 1

In [8]:

```
def grader_check_encoder():
    '''
        vocab-size: Unique words of the input language,
        embedding_size: output embedding dimension for each word after embedding layer,
        lstm_size: Number of lstm units,
        input_length: Length of the input sentence,
        batch_size
    '''
    vocab_size=10
    embedding_size=20
    lstm_size=32
    input_length=10
    batch_size=16
    #Intialzing encoder
    encoder=Encoder(vocab_size,embedding_size,lstm_size,input_length)
    input_sequence=tf.random.uniform(shape=[batch_size,input_length],maxval=vocab_size,
minval=0,dtype=tf.int32)
    #Intializing encoder initial states
    initial_state=encoder.initialize_states(batch_size)

    encoder_output,state_h,state_c=encoder(input_sequence,initial_state)

    assert(encoder_output.shape==(batch_size,input_length,lstm_size) and state_h.shape=
=(batch_size,lstm_size) and state_c.shape==(batch_size,lstm_size))
    return True
print(grader_check_encoder())
```

```
True
```

In [9]:

```
class Decoder(tf.keras.Model):
    '''
    Encoder model -- That takes a input sequence and returns output sequence
    '''
    def __init__(self,out_vocab_size,embedding_size,lstm_size,input_length):
        super().__init__()
        #Initialize Embedding layer
        self.dec_embed = Embedding(input_dim = out_vocab_size, output_dim = embedding_s
ize, input_length = input_length)
        #Intialize Decoder LSTM layer
        self.dec_lstm = LSTM(lstm_size, return_sequences = True, return_state = True)

    def call(self,input_sequence,initial_states):
        embedding = self.dec_embed(input_sequence)
        output_state, dec_h, dec_c = self.dec_lstm(embedding, initial_state = initial_s
tates)
        return output_state, dec_h, dec_c
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 2

In [10]:

```
def grader_decoder():
    '''
        out_vocab_size: Unique words of the target language,
        embedding_size: output embedding dimension for each word after embedding layer,
        dec_units: Number of lstm units in decoder,
        input_length: Length of the input sentence,
        batch_size


    '''
    out_vocab_size=13
    embedding_dim=12
    input_length=10
    dec_units=16
    batch_size=32

    target_sentences=tf.random.uniform(shape=(batch_size,input_length),maxval=10,minval
=0,dtype=tf.int32)
    encoder_output=tf.random.uniform(shape=[batch_size,input_length,dec_units])
    state_h=tf.random.uniform(shape=[batch_size,dec_units])
    state_c=tf.random.uniform(shape=[batch_size,dec_units])
    states=[state_h,state_c]
    decoder=Decoder(out_vocab_size, embedding_dim, dec_units,input_length )
    output,_,_=decoder(target_sentences, states)
    assert(output.shape==(batch_size,input_length,dec_units))
    return True
print(grader_decoder())
```

```
True
```

In [11]:

```
class Encoder_decoder(tf.keras.Model):
    def __init__(self,*params):
        super().__init__()
        #Create encoder object
        self.encoder = Encoder(inp_vocab_size = params[0], embedding_size = params[2],
lstm_size = params[3], input_length = params[4])
        #Create decoder object
        self.decoder = Decoder(out_vocab_size = params[1], embedding_size = params[2],
lstm_size = params[3], input_length = params[5])
        #Intialize Dense layer(out_vocab_size) with activation='softmax'
        self.dense = Dense(params[1], activation='softmax')

    @tf.function
    def call(self, params, training = True):

        enc_inp, dec_inp = params[0], params[1]
        initial_state = self.encoder.initialize_states(batch_size)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        output, _, _ = self.decoder(dec_inp ,[enc_h, enc_c])
        return self.dense(output)
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

In [12]:

```
data_preprocessing['english_inp'] = '<sos> '+data_preprocessing['english']
data_preprocessing['english_out'] = data_preprocessing['english'] + ' <eos>'
data_preprocessing['italin'] = data_preprocessing['italin'].apply(lambda x: str(x))
data_preprocessing['italin'] = '<sos> '+data_preprocessing['italin']+' <eos>'
```

In [ ]:

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(data_preprocessing, test_size = 0.1, random_state = 0)
train, validation = train_test_split(train, test_size = 0.1, random_state = 0)
```

In [13]:

```
from tensorflow.keras.preprocessing.text import Tokenizer
tokenizer_italian = Tokenizer()
tokenizer_italian.fit_on_texts(train['italin'].values)
tokenizer_english = Tokenizer(filters='!"#$%&()*+,-./:;=?@[\\]^_`{|}~\t\n')
tokenizer_english.fit_on_texts(train['english_inp'].values + train['english_out'].value
s)
```

In [14]:

```
eng_vc_len = data_preprocessing['italin'].astype(str).apply(lambda x: len(x))
ita_vc_len = data_preprocessing['english_inp'].astype(str).apply(lambda x: len(x))
```

In [15]:

```
for i in range(0,101,10):
    print(i,np.percentile(eng_vc_len, i))
for i in range(90,101):
    print(i,np.percentile(eng_vc_len, i))
for i in [99.1,99.2,99.3,99.4,99.5,99.6,99.7,99.8,99.9,100]:
    print(i,np.percentile(eng_vc_len, i))
```

```
0 12.0
10 24.0
20 26.0
30 28.0
40 29.0
50 31.0
60 32.0
70 34.0
80 35.0
90 38.0
100 112.0
90 38.0
91 38.0
92 39.0
93 39.0
94 40.0
95 40.0
96 41.0
97 42.0
98 43.0
99 45.0
100 112.0
99.1 45.0
99.2 46.0
99.3 46.0
99.4 46.0
99.5 47.0
99.6 48.0
99.7 48.0
99.8 49.0
99.9 51.00100000000384
100 112.0
```

In [16]:

```
for i in range(0,101,10):
    print(i,np.percentile(ita_vc_len, i))
for i in range(90,101):
    print(i,np.percentile(ita_vc_len, i))
for i in [99.1,99.2,99.3,99.4,99.5,99.6,99.7,99.8,99.9,100]:
    print(i,np.percentile(ita_vc_len, i))
```

```
0 8.0
10 18.0
20 20.0
30 21.0
40 22.0
50 23.0
60 24.0
70 25.0
80 25.0
90 26.0
100 43.0
90 26.0
91 26.0
92 26.0
93 26.0
94 26.0
95 27.0
96 27.0
97 27.0
98 27.0
99 28.0
100 43.0
99.1 28.0
99.2 28.0
99.3 28.0
99.4 28.0
99.5 28.0
99.6 28.0
99.7 28.0
99.8 29.0
99.9 29.0
100 43.0
```

In [17]:

```
eng_voc_len = 0
set_eng = set()
for i in data_preprocessing['english_inp'].values+data_preprocessing['english_out'].val
ues:
    x = len(i.split())
    if x > eng_voc_len:
        eng_voc_len = x
    for j in i.split():
        set_eng.add(j)
set_eng_size = len(set_eng)

ita_voc_len = 0
set_ita = set()
for i in data_preprocessing['italin']:
    x = len(i.split())
    if x > ita_voc_len:
        ita_voc_len = x
    for j in i.split():
        set_ita.add(j)
set_ita_size = len(set_ita)
```

In [18]:

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical

"""Ref: https://gist.github.com/ashishthomaschempolil/3539a83449645391328e4694b177b18
a"""
class Dataset:
    def __init__(self, data, tokenizer_italian, tokenizer_english, ita_voc_len, eng_voc
_len):
        self.encd_inputs = data['italin'].values
        self.decd_inputs = data['english_inp'].values
        self.decd_outputs = data['english_out'].values
        self.tokenizer_english = tokenizer_english
        self.tokenizer_italian = tokenizer_italian
        self.ita_voc_len = ita_voc_len
        self.eng_voc_len = eng_voc_len

    def __getitem__(self, i):
        self.encd_sequence = self.tokenizer_italian.texts_to_sequences([self.encd_input
s[i]]) # need to pass list of values
        self.decd_input_sequence = self.tokenizer_english.texts_to_sequences([self.decd
_inputs[i]])
        self.decd_output_sequence = self.tokenizer_english.texts_to_sequences([self.dec
d_outputs[i]])

        self.encd_sequence = pad_sequences(self.encd_sequence, maxlen=self.ita_voc_len,
dtype='int32', padding='post')
        self.decd_input_sequence = pad_sequences(self.decd_input_sequence, maxlen=self.
eng_voc_len, dtype='int32', padding='post')
        self.decd_output_sequence = pad_sequences(self.decd_output_sequence, maxlen=sel
f.eng_voc_len, dtype='int32', padding='post')
        return self.encd_sequence, self.decd_input_sequence, self.decd_output_sequence

    def __len__(self):
        return len(self.encd_inputs)

"""Ref: https://gist.github.com/ashishthomaschempolil/60277e4ca6e7541dc96ef195fd5b839
e"""
class Dataloder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size):
        self.dataset = dataset
        self.batch_size = batch_size
        self.indexes = np.arange(len(self.dataset.encd_inputs))

    def __getitem__(self, i):
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.squeeze(np.stack(samples, axis=1), axis=0) for samples in zip(*data
)]
        return tuple([[tf.convert_to_tensor(batch[0]), tf.convert_to_tensor(batch[1])],
tf.convert_to_tensor(batch[2])])

    def __len__(self):
        return len(self.indexes) // self.batch_size
    def on_epoch_end(self):
        self.indexes = np.random.permutation(self.indexes)
```

In [19]:

```
"""Ref: https://ppasumarthi-69210.medium.com/word-embeddings-in-keras-be6bb3092831"""
embeddings_index = dict()
f = open('glove.6B.300d.txt', encoding = 'utf-8')
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))
```

```
Loaded 400000 word vectors.
```

In [20]:

```
"""Ref: https://ppasumarthi-69210.medium.com/word-embeddings-in-keras-be6bb3092831"""
embedding_matrix = np.zeros((eng_vocab_size, 300))
for word, i in tokenizer_english.word_index.items():
    embedding_vector = embeddings_index.get(ord)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

In [21]:

```
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard, EarlyStopping, Red
uceLROnPlateau
import os
import datetime
batch_size=32
lstm_size=128
eng_voc_len = 50
ita_txt_len = 29
embedding_dim = 300
dense_units = 256

tr_dat = Dataset(train, tokenizer_italian, tokenizer_english, ita_txt_len, eng_voc_len)
te_dat  = Dataset(test, tokenizer_italian, tokenizer_english, ita_txt_len, eng_voc_len)
val_dat  = Dataset(validation, tokenizer_italian, tokenizer_english, ita_txt_len, eng_v
oc_len)

train_dataloader = Dataloder(tr_dat, batch_size=batch_size)
test_dataloader = Dataloder(te_dat, batch_size=batch_size)
val_dataloader = Dataloder(val_dat, batch_size = batch_size)
```

In [23]:

```
model = Encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_size, ita_txt_l
en, eng_voc_len, dense_units)
model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy')
log_dir="logs/seq2seq/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
call_back = [ModelCheckpoint('ita_eng_1', save_best_only= True, verbose = 1),TensorBoar
d(log_dir = log_dir,
                              histogram_freq=1, write_graph=True), EarlyStopping(patien
ce = 5, verbose = 1),
                              ReduceLROnPlateau(patience = 3, verbose = 1)]

model.fit(x = train_dataloader, steps_per_epoch = train_dataloader.__len__(),
          validation_data = val_dataloader, validation_steps = val_dataloader.__len__
(),
          epochs = 35, verbose = 1, callbacks = call_back)
```

```
Epoch 1/35
2531/2531 [==============================] - 275s 108ms/step - loss: 0.999
2 - val_loss: 0.3312

Epoch 00001: val_loss improved from inf to 0.33121, saving model to ita_en
g_1
Epoch 2/35
2531/2531 [==============================] - 273s 108ms/step - loss: 0.316
4 - val_loss: 0.2709

Epoch 00002: val_loss improved from 0.33121 to 0.27088, saving model to it
a_eng_1
Epoch 3/35
2531/2531 [==============================] - 278s 110ms/step - loss: 0.251
4 - val_loss: 0.2185

Epoch 00003: val_loss improved from 0.27088 to 0.21847, saving model to it
a_eng_1
Epoch 4/35
2531/2531 [==============================] - 278s 110ms/step - loss: 0.198
7 - val_loss: 0.1833

Epoch 00004: val_loss improved from 0.21847 to 0.18326, saving model to it
a_eng_1
Epoch 5/35
2531/2531 [==============================] - 278s 110ms/step - loss: 0.159
4 - val_loss: 0.1531

Epoch 00005: val_loss improved from 0.18326 to 0.15313, saving model to it
a_eng_1
Epoch 6/35
2531/2531 [==============================] - 279s 110ms/step - loss: 0.124
7 - val_loss: 0.1276

Epoch 00006: val_loss improved from 0.15313 to 0.12762, saving model to it
a_eng_1
Epoch 7/35
2531/2531 [==============================] - 279s 110ms/step - loss: 0.094
6 - val_loss: 0.1081

Epoch 00007: val_loss improved from 0.12762 to 0.10805, saving model to it
a_eng_1
Epoch 8/35
2531/2531 [==============================] - 279s 110ms/step - loss: 0.071
6 - val_loss: 0.0935

Epoch 00008: val_loss improved from 0.10805 to 0.09355, saving model to it
a_eng_1
Epoch 9/35
2531/2531 [==============================] - 279s 110ms/step - loss: 0.054
5 - val_loss: 0.0841

Epoch 00009: val_loss improved from 0.09355 to 0.08409, saving model to it
a_eng_1
Epoch 10/35
2531/2531 [==============================] - 277s 110ms/step - loss: 0.043
2 - val_loss: 0.0772

Epoch 00010: val_loss improved from 0.08409 to 0.07716, saving model to it
a_eng_1
Epoch 11/35
```

```
2531/2531 [==============================] - 277s 109ms/step - loss: 0.034
9 - val_loss: 0.0735

Epoch 00011: val_loss improved from 0.07716 to 0.07353, saving model to it
a_eng_1
Epoch 12/35
2531/2531 [==============================] - 276s 109ms/step - loss: 0.029
3 - val_loss: 0.0708

Epoch 00012: val_loss improved from 0.07353 to 0.07081, saving model to it
a_eng_1
Epoch 13/35
2531/2531 [==============================] - 275s 109ms/step - loss: 0.024
9 - val_loss: 0.0688

Epoch 00013: val_loss improved from 0.07081 to 0.06884, saving model to it
a_eng_1
Epoch 14/35
2531/2531 [==============================] - 271s 107ms/step - loss: 0.021
8 - val_loss: 0.0674

Epoch 00014: val_loss improved from 0.06884 to 0.06737, saving model to it
a_eng_1
Epoch 15/35
2531/2531 [==============================] - 270s 107ms/step - loss: 0.019
4 - val_loss: 0.0672

Epoch 00015: val_loss improved from 0.06737 to 0.06724, saving model to it
a_eng_1
Epoch 16/35
2531/2531 [==============================] - 272s 108ms/step - loss: 0.017
4 - val_loss: 0.0665

Epoch 00016: val_loss improved from 0.06724 to 0.06654, saving model to it
a_eng_1
Epoch 17/35
2531/2531 [==============================] - 270s 107ms/step - loss: 0.016
0 - val_loss: 0.0662

Epoch 00017: val_loss improved from 0.06654 to 0.06616, saving model to it
a_eng_1
Epoch 18/35
2531/2531 [==============================] - 274s 108ms/step - loss: 0.015
0 - val_loss: 0.0659

Epoch 00018: val_loss improved from 0.06616 to 0.06593, saving model to it
a_eng_1
Epoch 19/35
2531/2531 [==============================] - 272s 107ms/step - loss: 0.014
1 - val_loss: 0.0663

Epoch 00019: val_loss did not improve from 0.06593
Epoch 20/35
2531/2531 [==============================] - 272s 107ms/step - loss: 0.013
2 - val_loss: 0.0657

Epoch 00020: val_loss improved from 0.06593 to 0.06566, saving model to it
a_eng_1
Epoch 21/35
2531/2531 [==============================] - 271s 107ms/step - loss: 0.012
6 - val_loss: 0.0666
```

```
Epoch 00021: val_loss did not improve from 0.06566
Epoch 22/35
2531/2531 [==============================] - 270s 107ms/step - loss: 0.012
1 - val_loss: 0.0668

Epoch 00022: val_loss did not improve from 0.06566
Epoch 23/35
2531/2531 [==============================] - 272s 108ms/step - loss: 0.011
7 - val_loss: 0.0669

Epoch 00023: val_loss did not improve from 0.06566

Epoch 00023: ReduceLROnPlateau reducing learning rate to 0.000100000004749
74513.
Epoch 24/35
2531/2531 [==============================] - 266s 105ms/step - loss: 0.009
5 - val_loss: 0.0640

Epoch 00024: val_loss improved from 0.06566 to 0.06402, saving model to it
a_eng_1
Epoch 25/35
2531/2531 [==============================] - 266s 105ms/step - loss: 0.008
0 - val_loss: 0.0637

Epoch 00025: val_loss improved from 0.06402 to 0.06373, saving model to it
a_eng_1
Epoch 26/35
2531/2531 [==============================] - 269s 106ms/step - loss: 0.007
6 - val_loss: 0.0639

Epoch 00026: val_loss did not improve from 0.06373
Epoch 27/35
2531/2531 [==============================] - 270s 107ms/step - loss: 0.007
4 - val_loss: 0.0639

Epoch 00027: val_loss did not improve from 0.06373
Epoch 28/35
2531/2531 [==============================] - 271s 107ms/step - loss: 0.007
3 - val_loss: 0.0642

Epoch 00028: val_loss did not improve from 0.06373

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.000000047497451
4e-05.
Epoch 29/35
2531/2531 [==============================] - 271s 107ms/step - loss: 0.007
0 - val_loss: 0.0642

Epoch 00029: val_loss did not improve from 0.06373
Epoch 30/35
2531/2531 [==============================] - 270s 107ms/step - loss: 0.006
9 - val_loss: 0.0643

Epoch 00030: val_loss did not improve from 0.06373
Epoch 00030: early stopping
```

Out[23]:

```
<tensorflow.python.keras.callbacks.History at 0x7fcf0a031e80>
```

In [24]:

```
class pred_Encoder_decoder(tf.keras.Model):
    def __init__(self,*params):
        super().__init__()
        self.encoder = Encoder(inp_vocab_size = params[0], embedding_size = params[2],
lstm_size = params[3], input_length = params[4])
        self.decoder = Decoder(out_vocab_size = params[1], embedding_size = params[2],
lstm_size = params[3], input_length = params[5])
        self.dense = Dense(params[1], activation='softmax')

    def call(self, params, training = True):
        enc_inp = params[0]
        initial_state = self.encoder.initialize_states(1)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        pred = tf.expand_dims([tokenizer_english.word_index['<sos>']], 0)
        dec_h = enc_h
        dec_c = enc_c
        all_pred = []
        for t in range(eng_voc_len):
            pred, dec_h,dec_c = self.decoder(pred, [dec_h, dec_c])
            pred = self.dense(pred)
            pred = tf.argmax(pred, axis = -1)
            all_pred.append(pred)
        return all_pred
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

In [25]:

```
final_output = pred_Encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_siz
e, ita_txt_len, eng_voc_len, dense_units)
final_output.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy')
final_output.load_weights('/content/ita_eng_1')
```

Out[25]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcf0b
db2e80>
```

In [26]:

```
def predict(input_sequence):
    x = preprocess(input_sequence)
    x = '<sos> '+x+' <eos>'
    x = tokenizer_italian.texts_to_sequences([x])
    x = pad_sequences(seq, maxlen=ita_txt_len, padding='post', dtype = np.int32)
    y = final_output.predict(tf.expand_dims(seq, 0))
    z = []
    for i in y:
        word = tokenizer_english.index_word[i[0][0]]
        if word == '<eos>':
            break
        z.append(word)
    return ' '.join(z)
```

In [27]:

```
pr_final = train['italin'].values[0][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[0])
```

```
input :  ha appena chiamato qualcuno
predicted output :  somebody just called
actual output : somebody just called
```

In [28]:

```
pr_final = train['italin'].values[1000][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[1000])
```

```
input :  io ero gelosa di voi
predicted output :  i was jealous of you
actual output : i was jealous of you
```

In [29]:

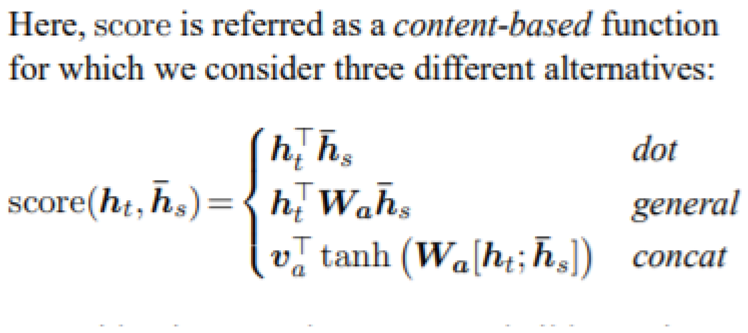
```
from nltk.translate.bleu_score import sentence_bleu
initial =0
for i in range(1000):
    pr_final = test['italin'].values[i][6:-6]
    con = test['english'].values[i]
    pred = predict(pr_final)
    initial+= sentence_bleu([con.split()], pred.split())
print('Bleu Score : {}'.format(score/1000))
```

```
Bleu Score : 0.8391122289889017
```

Task -2: Including Attention mechanisum

1. Use the preprocessed data from Task-1
2. You have to implement an Encoder and Decoder architecture with attention as discussed in the reference notebook.

   - Encoder - with 1 layer LSTM
   - Decoder - with 1 layer LSTM
   - attention - (Please refer the <a href= 'https://drive.google.com/file/d/1z_bnc-3aubKawbR6q8wyI6Mh5ho2R1aZ/view?usp=sharing'>**reference (https://drive.google.com/file/d/1z_bnc-3aubKawbR6q8wyI6Mh5ho2R1aZ/view?usp=sharing'>**reference) notebook**</a> to know more about the attention mechanism.)
3. In Global attention, we have 3 types of scoring functions(as discussed in the reference notebook). As a part of this assignment **you need to create 3 models for each scoring function**



Here, score is referred as a *content-based* function for which we consider three different alternatives:

$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

   - In model 1 you need to implemnt "dot" score function
   - In model 2 you need to implemnt "general" score function
   - In model 3 you need to implemnt "concat" score function.

   Please do add the markdown titles for each model so that we can have a better look at the code and verify.
4. It is mandatory to train the model with simple model.fit() only, Donot train the model with custom GradientTape()
5. Using attention weights, you can plot the attention plots, please plot those for 2-3 examples. You can check about those in this (https://www.tensorflow.org/tutorials/text/nmt_with_attention#translate)
6. The attention layer has to be written by yourself only. The main objective of this assignment is to read and implement a paper on yourself so please do it yourself.
7. Please implement the class **onestepdecoder** as mentioned in the assignment instructions.
8. You can use any tf.Keras highlevel API's to build and train the models. Check the reference notebook for better understanding.
9. Use BLEU score as metric to evaluate your model. You can use any loss function you need.
10. You have to use Tensorboard to plot the Graph, Scores and histograms of gradients.
11. Resources: a. Check the reference notebook b. Resource 1 (https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/) c. Resource 2 (https://www.tensorflow.org/tutorials/text/nmt_with_attention) d. Resource 3 (https://stackoverflow.com/questions/44238154/what-is-the-difference-between-luong-attention-and-bahdanau-attention#:~:text=Luong%20attention%20used%20top%20hidden,hidden%20state%20at%20time%20t.)

Implement custom encoder decoder and attention layers

Encoder

In [30]:

```
from tensorflow.keras.layers import *
class Encoder(tf.keras.Model):

    def __init__(self,inp_vocab_size,embedding_size,lstm_size,input_length):
        super(Encoder, self).__init__()
        self.lstm_size = lstm_size
        self.enc_embed = Embedding(input_dim = inp_vocab_size, output_dim = embedding_s
ize)
        self.enc_lstm = LSTM(lstm_size, return_sequences = True, return_state = True)

    def call(self,input_sequence,states):
        embedding = self.enc_embed(input_sequence)
        output_state, enc_h, enc_c = self.enc_lstm(embedding, initial_state = states)
        return output_state, enc_h, enc_c

    def initialize_states(self,batch_size):
        return [tf.zeros((batch_size, self.lstm_size)), tf.zeros((batch_size, self.lstm
_size))]
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 1

In [31]:

```
def grader_check_encoder():

    '''
        vocab-size: Unique words of the input language,
        embedding_size: output embedding dimension for each word after embedding layer,
        lstm_size: Number of lstm units in encoder,
        input_length: Length of the input sentence,
        batch_size
    '''


    vocab_size=10
    embedding_size=20
    lstm_size=32
    input_length=10
    batch_size=16
    encoder=Encoder(vocab_size,embedding_size,lstm_size,input_length)
    input_sequence=tf.random.uniform(shape=[batch_size,input_length],maxval=vocab_size,
minval=0,dtype=tf.int32)
    initial_state=encoder.initialize_states(batch_size)
    encoder_output,state_h,state_c=encoder(input_sequence,initial_state)

    assert(encoder_output.shape==(batch_size,input_length,lstm_size) and state_h.shape=
=(batch_size,lstm_size) and state_c.shape==(batch_size,lstm_size))
    return True
print(grader_check_encoder())
```

```
True
```

Attention

In [32]:

```
from tensorflow.keras.layers import *
class Attention(tf.keras.layers.Layer):

    def __init__(self,scoring_function, att_units):
        super(Attention, self).__init__()
        self.scoring_function = scoring_function
        if scoring_function == 'dot':
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'general':
            self.W = Dense(att_units)
            self.dot = Dot(axes = (1, 2))
        elif scoring_function == 'concat':
            self.W1 = Dense(att_units)
            self.W2 = Dense(att_units)
            self.V = Dense(1)
    def call(self,decoder_hidden_state,encoder_output):

        decoder_hidden_state = tf.expand_dims(decoder_hidden_state, 1)

        if self.scoring_function == 'dot':
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state, (0, 2, 1
)), encoder_output]), (0, 2,1))

        elif self.scoring_function == 'general':
            mul = self.W(encoder_output)
            score = tf.transpose(self.dot([tf.transpose(decoder_hidden_state, (0, 2, 1
)), mul]), (0, 2,1))

        elif self.scoring_function == 'concat':
            inter = self.W1(decoder_hidden_state) + self.W2(encoder_output)
            tan = tf.nn.tanh(inter)
            score = self.V(tan)
        attention_weights = tf.nn.softmax(score, axis =1)
        context_vector = attention_weights * encoder_output
        context_vector = tf.reduce_sum(context_vector, axis=1)
        return context_vector, attention_weights
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 2

In [33]:

```
def grader_check_attention(scoring_fun):

    '''
        att_units: Used in matrix multiplications for scoring functions,
        input_length: Length of the input sentence,
        batch_size
    '''

    input_length=10
    batch_size=16
    att_units=32

    state_h=tf.random.uniform(shape=[batch_size,att_units])
    encoder_output=tf.random.uniform(shape=[batch_size,input_length,att_units])
    attention=Attention(scoring_fun,att_units)
    context_vector,attention_weights=attention(state_h,encoder_output)
    assert(context_vector.shape==(batch_size,att_units) and attention_weights.shape==(b
atch_size,input_length,1))
    return True
print(grader_check_attention('dot'))
print(grader_check_attention('general'))
print(grader_check_attention('concat'))
```

```
True
True
True
```

OneStepDecoder

In [34]:

```
class OneStepDecoder(tf.keras.layers.Layer):
    def __init__(self,tar_vocab_size, embedding_dim, input_length, dec_units ,score_fun
,att_units):
        super(OneStepDecoder, self).__init__()
        self.embed_dec = Embedding(input_dim = tar_vocab_size, output_dim = embedding_d
im)
        self.lstm = LSTM(dec_units, return_sequences = True, return_state = True)
        self.attention = Attention(scoring_function = score_fun, att_units = att_units)
        self.fc = Dense(tar_vocab_size)

    def call(self,input_to_decoder, encoder_output, state_h,state_c):
        embed = self.embed_dec(input_to_decoder)
        context_vect, attention_weights = self.attention(state_h, encoder_output)
        final_inp = tf.concat([tf.expand_dims(context_vect, 1), embed], axis = -1)
        out, dec_h, dec_c = self.lstm(final_inp, [state_h, state_c])
        out = tf.reshape(out, (-1, out.shape[2]))
        output = self.fc(out)
        return output, dec_h, dec_c, attention_weights, context_vect
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 3

In [35]:

```
def grader_onestepdecoder(score_fun):

    tar_vocab_size=13
    embedding_dim=12
    input_length=10
    dec_units=16
    att_units=16
    batch_size=32
    onestepdecoder=OneStepDecoder(tar_vocab_size, embedding_dim, input_length, dec_units ,score_fun ,att_units)
    input_to_decoder=tf.random.uniform(shape=(batch_size,1),maxval=10,minval=0,dtype=tf.int32)
    encoder_output=tf.random.uniform(shape=[batch_size,input_length,dec_units])
    state_h=tf.random.uniform(shape=[batch_size,dec_units])
    state_c=tf.random.uniform(shape=[batch_size,dec_units])
    output,state_h,state_c,attention_weights,context_vector=onestepdecoder(input_to_decoder,encoder_output,state_h,state_c)
    assert(output.shape==(batch_size,tar_vocab_size))
    assert(state_h.shape==(batch_size,dec_units))
    assert(state_c.shape==(batch_size,dec_units))
    assert(attention_weights.shape==(batch_size,input_length,1))
    assert(context_vector.shape==(batch_size,dec_units))
    return True

print(grader_onestepdecoder('dot'))
print(grader_onestepdecoder('general'))
print(grader_onestepdecoder('concat'))
```

```
True
True
True
```

Decoder

In [36]:

```
class Decoder(tf.keras.layers.Layer):
    def __init__(self,out_vocab_size, embedding_dim, input_length, dec_units ,score_fun
,att_units):
        super(Decoder, self).__init__()
        self.input_length = input_length
        self.out_vocab_size = out_vocab_size
        self.one_step_decoder = OneStepDecoder(out_vocab_size, embedding_dim, input_len
gth, dec_units ,score_fun ,att_units)
        self.out_vocab_size = out_vocab_size

    def call(self, input_to_decoder, encoder_output, decoder_hidden_state, decoder_cell
_state):
        all_outputs = tf.TensorArray(dtype = tf.float32, size= input_to_decoder.shape[1
])

        for timestep in range(input_to_decoder.shape[1]):
            output, decoder_hidden_state, decoder_cell_state, _, _ = self.one_step_deco
der(input_to_decoder[:, timestep:timestep+1],

encoder_output,

decoder_hidden_state,

decoder_cell_state)
            all_outputs = all_outputs.write(timestep, output)
        all_outputs = tf.transpose(all_outputs.stack(), (1, 0, 2))
        return all_outputs
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Grader function - 4

In [37]:

```
def grader_decoder(score_fun):

    '''
        out_vocab_size: Unique words of the target language,
        embedding_dim: output embedding dimension for each word after embedding layer,
        dec_units: Number of lstm units in decoder,
        att_units: Used in matrix multiplications for scoring functions in attention class,
        input_length: Length of the target sentence,
        batch_size


    '''

    out_vocab_size=13
    embedding_dim=12
    input_length=11
    dec_units=16
    att_units=16
    batch_size=32

    target_sentences=tf.random.uniform(shape=(batch_size,input_length),maxval=10,minval=0,dtype=tf.int32)
    encoder_output=tf.random.uniform(shape=[batch_size,input_length,dec_units])
    state_h=tf.random.uniform(shape=[batch_size,dec_units])
    state_c=tf.random.uniform(shape=[batch_size,dec_units])

    decoder=Decoder(out_vocab_size, embedding_dim, input_length, dec_units ,score_fun , att_units)
    output=decoder(target_sentences,encoder_output, state_h, state_c)
    assert(output.shape==(batch_size,input_length,out_vocab_size))
    return True
print(grader_decoder('dot'))
print(grader_decoder('general'))
print(grader_decoder('concat'))
```

```
True
True
True
```

Encoder Decoder model

In [38]:

```
class encoder_decoder(tf.keras.Model):
    def __init__(self, inp_vocab_size, out_vocab_size, embedding_dim, enc_units, dec_un
its, max_len_ita, max_len_eng, score_fun, att_units, batch_size):
        super(encoder_decoder, self).__init__()
        self.encoder = Encoder(inp_vocab_size, embedding_dim, enc_units, max_len_ita)
        self.OneStepDecoder = OneStepDecoder(out_vocab_size, embedding_dim, max_len_eng
, dec_units ,score_fun ,att_units)
        self.batch_size = batch_size

    @tf.function
    def call(self, data):
        enc_inp, dec_inp = data[0], data[1]
        initial_state = self.encoder.initialize_states(self.batch_size)
        enc_output, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        all_outputs = tf.TensorArray(dtype = tf.float32, size= 50)

        dec_h = enc_h
        dec_c = enc_c
        for timestep in range(50):
            output, dec_h, dec_c, _, _ = self.OneStepDecoder(dec_inp[:, timestep:timest
ep+1],
                                                               enc_output,
                                                               dec_h,
                                                               dec_c)
            all_outputs = all_outputs.write(timestep, output)
        all_outputs = tf.transpose(all_outputs.stack(), (1, 0, 2))
        return all_outputs
```

Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c (https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c)

Custom loss function

In [39]:

```
# Ref: https://www.tensorflow.org/tutorials/text/nmt_with_attention
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True, reduction
='none')
def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_mean(loss_)
```

Training

- Implement dot function here.

In [40]:

```
from tensorflow.keras.callbacks import*
import os
batch_size=128
lstm_size=128
max_len_eng = 50
ita_txt_len = 29
embedding_dim = 100
att_units = 256

tr_dat = Dataset(train, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
te_dat  = Dataset(test, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
val_dat  = Dataset(validation, tokenizer_italian, tokenizer_english, ita_txt_len, max_l
en_eng)

train_dataloader = Dataloder(tr_dat, batch_size=batch_size)
test_dataloader = Dataloder(te_dat, batch_size=batch_size)
val_dataloader = Dataloder(val_dat, batch_size = batch_size)
```

In [41]:

```
model = encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_size, lstm_size
,
                        ita_txt_len, max_len_eng, 'dot', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)
log_dir="logs/seq2seq/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
call_back = [ModelCheckpoint('ita_eng_2', save_best_only= True, verbose = 1), TensorBoa
rd(log_dir = log_dir,
                             histogram_freq=1, write_graph=True), EarlyStopping(patienc
e = 5, verbose = 1), ReduceLROnPlateau(patience = 3,

verbose = 1)]
```

In [42]:

```
model.fit(x = train_dataloader, steps_per_epoch = train_dataloader.__len__(), validation_data = val_dataloader,
          validation_steps = val_dataloader.__len__(), epochs = 35, verbose = 1, callbacks = call_back)
```

```
Epoch 1/35
632/632 [==============================] - 170s 202ms/step - loss: 0.5416
- val_loss: 0.3954

Epoch 00001: val_loss improved from inf to 0.39538, saving model to ita_en
g_2
Epoch 2/35
632/632 [==============================] - 115s 183ms/step - loss: 0.3848
- val_loss: 0.3417

Epoch 00002: val_loss improved from 0.39538 to 0.34175, saving model to it
a_eng_2
Epoch 3/35
632/632 [==============================] - 114s 181ms/step - loss: 0.3314
- val_loss: 0.3047

Epoch 00003: val_loss improved from 0.34175 to 0.30474, saving model to it
a_eng_2
Epoch 4/35
632/632 [==============================] - 114s 180ms/step - loss: 0.2943
- val_loss: 0.2750

Epoch 00004: val_loss improved from 0.30474 to 0.27497, saving model to it
a_eng_2
Epoch 5/35
632/632 [==============================] - 114s 181ms/step - loss: 0.2613
- val_loss: 0.2449

Epoch 00005: val_loss improved from 0.27497 to 0.24493, saving model to it
a_eng_2
Epoch 6/35
632/632 [==============================] - 115s 182ms/step - loss: 0.2302
- val_loss: 0.2198

Epoch 00006: val_loss improved from 0.24493 to 0.21983, saving model to it
a_eng_2
Epoch 7/35
632/632 [==============================] - 116s 184ms/step - loss: 0.2031
- val_loss: 0.1998

Epoch 00007: val_loss improved from 0.21983 to 0.19979, saving model to it
a_eng_2
Epoch 8/35
632/632 [==============================] - 116s 183ms/step - loss: 0.1806
- val_loss: 0.1834

Epoch 00008: val_loss improved from 0.19979 to 0.18336, saving model to it
a_eng_2
Epoch 9/35
632/632 [==============================] - 115s 182ms/step - loss: 0.1617
- val_loss: 0.1678

Epoch 00009: val_loss improved from 0.18336 to 0.16776, saving model to it
a_eng_2
Epoch 10/35
632/632 [==============================] - 114s 181ms/step - loss: 0.1428
- val_loss: 0.1536

Epoch 00010: val_loss improved from 0.16776 to 0.15365, saving model to it
a_eng_2
Epoch 11/35
```

```
632/632 [==============================] - 114s 181ms/step - loss: 0.1257
- val_loss: 0.1406

Epoch 00011: val_loss improved from 0.15365 to 0.14063, saving model to it
a_eng_2
Epoch 12/35
632/632 [==============================] - 114s 180ms/step - loss: 0.1097
- val_loss: 0.1287

Epoch 00012: val_loss improved from 0.14063 to 0.12873, saving model to it
a_eng_2
Epoch 13/35
632/632 [==============================] - 114s 181ms/step - loss: 0.0952
- val_loss: 0.1175

Epoch 00013: val_loss improved from 0.12873 to 0.11752, saving model to it
a_eng_2
Epoch 14/35
632/632 [==============================] - 114s 180ms/step - loss: 0.0821
- val_loss: 0.1085

Epoch 00014: val_loss improved from 0.11752 to 0.10854, saving model to it
a_eng_2
Epoch 15/35
632/632 [==============================] - 114s 181ms/step - loss: 0.0709
- val_loss: 0.1008

Epoch 00015: val_loss improved from 0.10854 to 0.10083, saving model to it
a_eng_2
Epoch 16/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0615
- val_loss: 0.0945

Epoch 00016: val_loss improved from 0.10083 to 0.09452, saving model to it
a_eng_2
Epoch 17/35
632/632 [==============================] - 116s 183ms/step - loss: 0.0528
- val_loss: 0.0896

Epoch 00017: val_loss improved from 0.09452 to 0.08957, saving model to it
a_eng_2
Epoch 18/35
632/632 [==============================] - 116s 184ms/step - loss: 0.0460
- val_loss: 0.0843

Epoch 00018: val_loss improved from 0.08957 to 0.08427, saving model to it
a_eng_2
Epoch 19/35
632/632 [==============================] - 115s 183ms/step - loss: 0.0407
- val_loss: 0.0808

Epoch 00019: val_loss improved from 0.08427 to 0.08080, saving model to it
a_eng_2
Epoch 20/35
632/632 [==============================] - 114s 181ms/step - loss: 0.0357
- val_loss: 0.0781

Epoch 00020: val_loss improved from 0.08080 to 0.07811, saving model to it
a_eng_2
Epoch 21/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0317
```

```
- val_loss: 0.0756

Epoch 00021: val_loss improved from 0.07811 to 0.07558, saving model to it
a_eng_2
Epoch 22/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0284
- val_loss: 0.0737

Epoch 00022: val_loss improved from 0.07558 to 0.07367, saving model to it
a_eng_2
Epoch 23/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0255
- val_loss: 0.0729

Epoch 00023: val_loss improved from 0.07367 to 0.07287, saving model to it
a_eng_2
Epoch 24/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0231
- val_loss: 0.0704

Epoch 00024: val_loss improved from 0.07287 to 0.07036, saving model to it
a_eng_2
Epoch 25/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0209
- val_loss: 0.0701

Epoch 00025: val_loss improved from 0.07036 to 0.07013, saving model to it
a_eng_2
Epoch 26/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0193
- val_loss: 0.0699

Epoch 00026: val_loss improved from 0.07013 to 0.06993, saving model to it
a_eng_2
Epoch 27/35
632/632 [==============================] - 115s 183ms/step - loss: 0.0177
- val_loss: 0.0688

Epoch 00027: val_loss improved from 0.06993 to 0.06882, saving model to it
a_eng_2
Epoch 28/35
632/632 [==============================] - 115s 182ms/step - loss: 0.0166
- val_loss: 0.0685

Epoch 00028: val_loss improved from 0.06882 to 0.06851, saving model to it
a_eng_2
Epoch 29/35
632/632 [==============================] - 115s 183ms/step - loss: 0.0153
- val_loss: 0.0681

Epoch 00029: val_loss improved from 0.06851 to 0.06808, saving model to it
a_eng_2
Epoch 30/35
632/632 [==============================] - 115s 181ms/step - loss: 0.0143
- val_loss: 0.0681

Epoch 00030: val_loss did not improve from 0.06808
Epoch 31/35
632/632 [==============================] - 114s 181ms/step - loss: 0.0134
- val_loss: 0.0680
```

```
Epoch 00031: val_loss improved from 0.06808 to 0.06798, saving model to it
a_eng_2
Epoch 32/35
632/632 [==============================] - 114s 181ms/step - loss: 0.0124
- val_loss: 0.0673

Epoch 00032: val_loss improved from 0.06798 to 0.06735, saving model to it
a_eng_2
Epoch 33/35
632/632 [==============================] - 115s 181ms/step - loss: 0.0117
- val_loss: 0.0681

Epoch 00033: val_loss did not improve from 0.06735
Epoch 34/35
632/632 [==============================] - 115s 181ms/step - loss: 0.0112
- val_loss: 0.0677

Epoch 00034: val_loss did not improve from 0.06735
Epoch 35/35
632/632 [==============================] - 114s 180ms/step - loss: 0.0107
- val_loss: 0.0680

Epoch 00035: val_loss did not improve from 0.06735

Epoch 00035: ReduceLROnPlateau reducing learning rate to 0.000100000004749
74513.
```

Out[42]:

```
<tensorflow.python.keras.callbacks.History at 0x7fce499d1860>
```

In [43]:

```python
# Ref: https://vineethaswani2.medium.com/spelling-error-correction-7154f781354c
class pred_Encoder_decoder(tf.keras.Model):
    def __init__(self, inp_vocab_size, out_vocab_size, embedding_dim, enc_units, dec_un
its, max_len_ita, max_len_eng, score_fun, att_units):
        super(pred_Encoder_decoder, self).__init__()
        self.encoder = Encoder(inp_vocab_size, embedding_dim, enc_units, max_len_ita)
        self.OneStepDecoder = OneStepDecoder(out_vocab_size, embedding_dim, max_len_eng
, dec_units ,score_fun ,att_units)
        self.batch_size = batch_size
    def call(self, params):
        enc_inp = params[0]
        initial_state = self.encoder.initialize_states(1)
        output_state, enc_h, enc_c = self.encoder(enc_inp, initial_state)
        pred = tf.expand_dims([tokenizer_english.word_index['<sos>']], 0)
        dec_h = enc_h
        dec_c = enc_c
        all_pred = []
        all_attention = []
        for t in range(50):
            pred, dec_h,dec_c, attention, _ = self.OneStepDecoder(pred, output_state, d
ec_h, dec_c)
            pred = tf.argmax(pred, axis = -1)
            all_pred.append(pred)
            pred = tf.expand_dims(pred, 0)
            all_attention.append(attention)
        return all_pred, all_attention
```

In [44]:

```
final_output = pred_Encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_siz
e, lstm_size, ita_txt_len, max_len_eng, 'dot', att_units)
final_output.compile(optimizer = 'Adam', loss = loss_function)
final_output.load_weights('/content/ita_eng_2')
```

Out[44]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcdc3
c59320>
```

Inference

Plot attention weights

In [45]:

```
# Ref: https://www.tensorflow.org/tutorials/text/nmt_with_attention
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker

def plot_attention(attention, sentence, predicted_sentence):
    fig = plt.figure(figsize=(12,12))
    ax = fig.add_subplot(1, 1, 1)
    ax.matshow(attention, cmap='YlOrRd')

    fontdict = {'fontsize': 14}

    ax.set_xticklabels([''] + sentence, fontdict=fontdict, rotation=90)
    ax.set_yticklabels([''] + predicted_sentence, fontdict=fontdict)

    ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
    ax.yaxis.set_major_locator(ticker.MultipleLocator(1))

    plt.show()
```

Predict the sentence translation

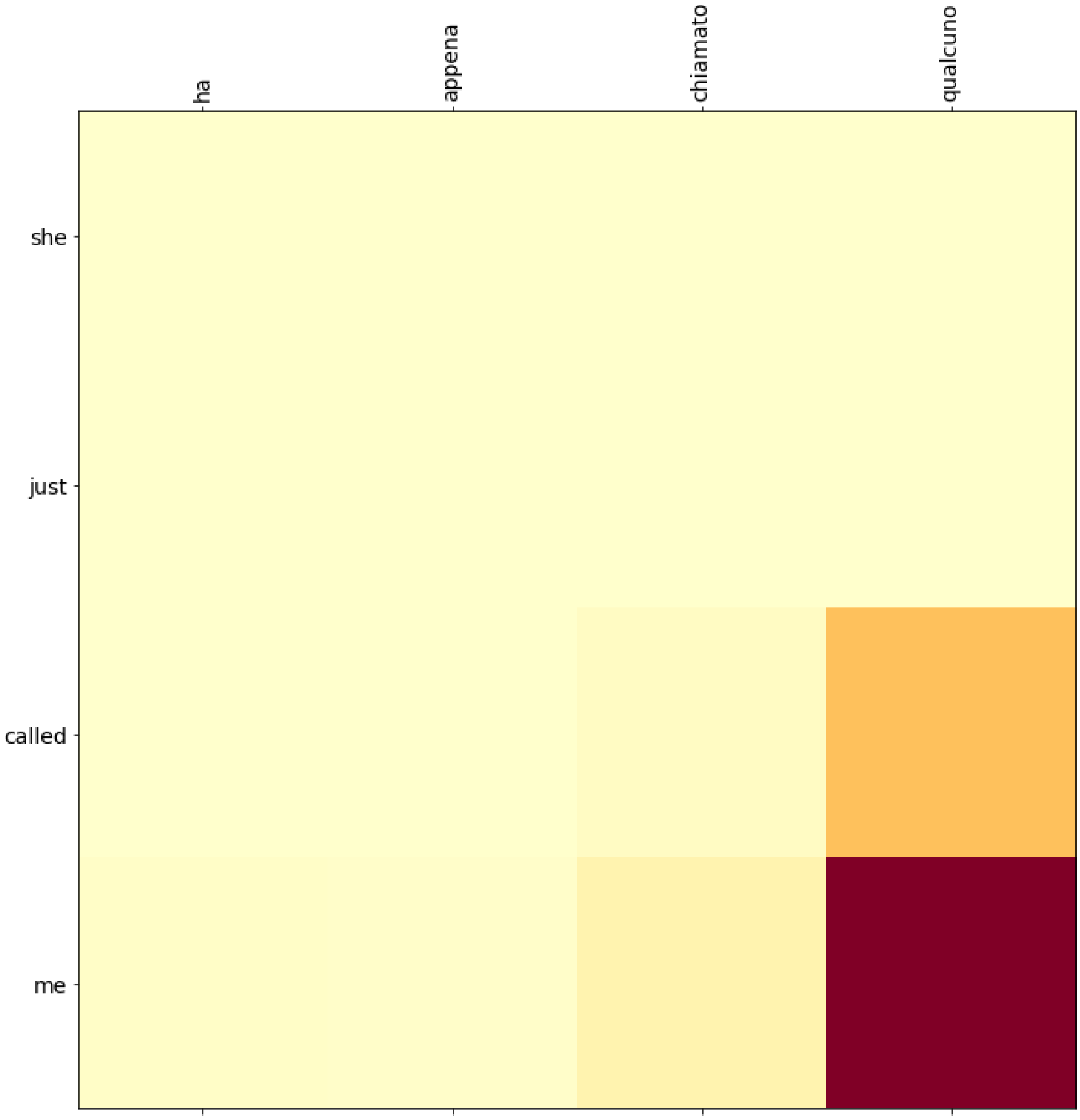
In [46]:

```
def predict(input_sequence):
    x = preprocess(input_sequence)
    x = '<sos> '+x+' <eos>'
    x = tokenizer_italian.texts_to_sequences([x])
    x = pad_sequences(seq, maxlen=ita_txt_len, padding='post', dtype = np.int32)
    y = final_output.predict(tf.expand_dims(seq, 0))
    z = []
    for i in y:
        word = tokenizer_english.index_word[i[0][0]]
        if word == '<eos>':
            break
        z.append(word)
    return ' '.join(z)
```

In [47]:

```
pr_final = train['italin'].values[0][6:-6]
print('input : ', pr_final)
result, attention_plot = predict(pr_final)
attention_plot = attention_plot[:len(result.split(' ')), :len(pr_final.split(' '))]
plot_attention(attention_plot, pr_final.split(' '), result.split(' '))
print('predicted output : ',result)
print('actual output :', train['english'].values[0])
```

```
input :  ha appena chiamato qualcuno
```

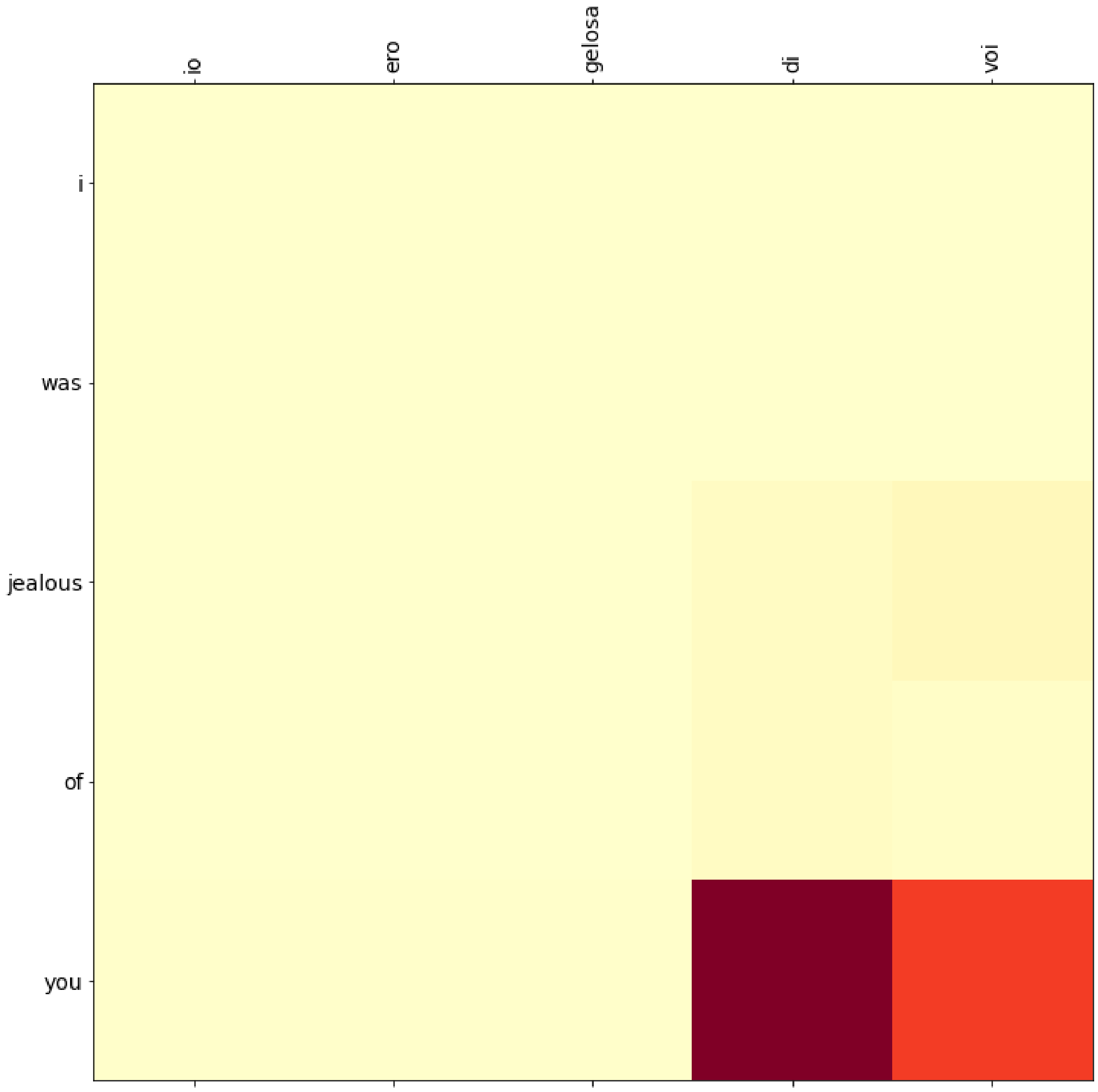


```
predicted output :  she just called me
actual output : somebody just called
```

In [48]:

```
pr_final = train['italin'].values[1000][6:-6]
print('input : ', pr_final)
result, attention_plot = predict(pr_final)
attention_plot = attention_plot[:len(result.split(' ')), :len(pr_final.split(' '))]
plot_attention(attention_plot, pr_final.split(' '), result.split(' '))
print('predicted output : ',result)
print('actual output :', train['english'].values[1000])
```

```
input :  io ero gelosa di voi
```



```
predicted output :  i was jealous of you
actual output : i was jealous of you
```

****Calculate BLEU score****

In [49]:

```
from nltk.translate.bleu_score import sentence_bleu
initial =0
for i in range(1000):
    pr_final = test['italin'].values[i][6:-6]
    con = test['english'].values[i]
    pred = predict(pr_final)
    initial+= sentence_bleu([con.split()], pred.split())
print('Bleu Score : {}'.format(score/1000))
```

```
Bleu Score : 0.8262435337297614
```

Repeat the same steps for General scoring function

In [50]:

```
from tensorflow.keras.callbacks import*
import os
import datetime

batch_size=128
lstm_size=256
embedding_dim = 100

tr_dat = Dataset(train, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
te_dat  = Dataset(test, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
val_dat  = Dataset(validation, tokenizer_italian, tokenizer_english, ita_txt_len, max_l
en_eng)

train_dataloader = Dataloder(tr_dat, batch_size=batch_size)
test_dataloader = Dataloder(te_dat, batch_size=batch_size)
val_dataloader = Dataloder(val_dat, batch_size = batch_size)
```

In [51]:

```
model = encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_size,
                        lstm_size, ita_txt_len, max_len_eng, 'general', att_units, batc
h_size)
model.compile(optimizer = 'Adam', loss = loss_function)
log_dir="logs/seq2seq/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S") call_back =
[ModelCheckpoint('ita_eng_3', save_best_only= True, verbose = 1),
             TensorBoard(log_dir = log_dir, histogram_freq=1, write_graph=True), EarlyS
topping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]
```

In [52]:

```
model.fit(x = train_dataloader, steps_per_epoch = train_dataloader.__len__(), validation_data = val_dataloader,
          validation_steps = val_dataloader.__len__(), epochs = 35, verbose = 1, callbacks = call_back)
```

```
Epoch 1/35
632/632 [==============================] - 212s 262ms/step - loss: 0.5123
- val_loss: 0.3987

Epoch 00001: val_loss improved from inf to 0.39868, saving model to ita_en
g_3
Epoch 2/35
632/632 [==============================] - 148s 235ms/step - loss: 0.3832
- val_loss: 0.3175

Epoch 00002: val_loss improved from 0.39868 to 0.31746, saving model to it
a_eng_3
Epoch 3/35
632/632 [==============================] - 149s 236ms/step - loss: 0.3014
- val_loss: 0.2587

Epoch 00003: val_loss improved from 0.31746 to 0.25873, saving model to it
a_eng_3
Epoch 4/35
632/632 [==============================] - 148s 235ms/step - loss: 0.2429
- val_loss: 0.2157

Epoch 00004: val_loss improved from 0.25873 to 0.21567, saving model to it
a_eng_3
Epoch 5/35
632/632 [==============================] - 149s 235ms/step - loss: 0.1979
- val_loss: 0.1817

Epoch 00005: val_loss improved from 0.21567 to 0.18166, saving model to it
a_eng_3
Epoch 6/35
632/632 [==============================] - 148s 234ms/step - loss: 0.1597
- val_loss: 0.1512

Epoch 00006: val_loss improved from 0.18166 to 0.15124, saving model to it
a_eng_3
Epoch 7/35
632/632 [==============================] - 149s 236ms/step - loss: 0.1263
- val_loss: 0.1283

Epoch 00007: val_loss improved from 0.15124 to 0.12835, saving model to it
a_eng_3
Epoch 8/35
632/632 [==============================] - 149s 236ms/step - loss: 0.0979
- val_loss: 0.1076

Epoch 00008: val_loss improved from 0.12835 to 0.10761, saving model to it
a_eng_3
Epoch 9/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0748
- val_loss: 0.0937

Epoch 00009: val_loss improved from 0.10761 to 0.09371, saving model to it
a_eng_3
Epoch 10/35
632/632 [==============================] - 150s 238ms/step - loss: 0.0581
- val_loss: 0.0831

Epoch 00010: val_loss improved from 0.09371 to 0.08306, saving model to it
a_eng_3
Epoch 11/35
```

```
632/632 [==============================] - 150s 238ms/step - loss: 0.0454
- val_loss: 0.0758

Epoch 00011: val_loss improved from 0.08306 to 0.07577, saving model to it
a_eng_3
Epoch 12/35
632/632 [==============================] - 150s 237ms/step - loss: 0.0358
- val_loss: 0.0706

Epoch 00012: val_loss improved from 0.07577 to 0.07058, saving model to it
a_eng_3
Epoch 13/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0293
- val_loss: 0.0669

Epoch 00013: val_loss improved from 0.07058 to 0.06689, saving model to it
a_eng_3
Epoch 14/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0245
- val_loss: 0.0643

Epoch 00014: val_loss improved from 0.06689 to 0.06434, saving model to it
a_eng_3
Epoch 15/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0205
- val_loss: 0.0633

Epoch 00015: val_loss improved from 0.06434 to 0.06329, saving model to it
a_eng_3
Epoch 16/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0175
- val_loss: 0.0617

Epoch 00016: val_loss improved from 0.06329 to 0.06170, saving model to it
a_eng_3
Epoch 17/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0156
- val_loss: 0.0611

Epoch 00017: val_loss improved from 0.06170 to 0.06109, saving model to it
a_eng_3
Epoch 18/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0139
- val_loss: 0.0617

Epoch 00018: val_loss did not improve from 0.06109
Epoch 19/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0126
- val_loss: 0.0605

Epoch 00019: val_loss improved from 0.06109 to 0.06052, saving model to it
a_eng_3
Epoch 20/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0114
- val_loss: 0.0602

Epoch 00020: val_loss improved from 0.06052 to 0.06024, saving model to it
a_eng_3
Epoch 21/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0104
- val_loss: 0.0608
```

```
Epoch 00021: val_loss did not improve from 0.06024
Epoch 22/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0101
- val_loss: 0.0600

Epoch 00022: val_loss improved from 0.06024 to 0.06004, saving model to it
a_eng_3
Epoch 23/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0092
- val_loss: 0.0608

Epoch 00023: val_loss did not improve from 0.06004
Epoch 24/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0088
- val_loss: 0.0619

Epoch 00024: val_loss did not improve from 0.06004
Epoch 25/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0085
- val_loss: 0.0613

Epoch 00025: val_loss did not improve from 0.06004

Epoch 00025: ReduceLROnPlateau reducing learning rate to 0.000100000004749
74513.
Epoch 26/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0066
- val_loss: 0.0589

Epoch 00026: val_loss improved from 0.06004 to 0.05885, saving model to it
a_eng_3
Epoch 27/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0053
- val_loss: 0.0589

Epoch 00027: val_loss did not improve from 0.05885
Epoch 28/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0050
- val_loss: 0.0590

Epoch 00028: val_loss did not improve from 0.05885
Epoch 29/35
632/632 [==============================] - 151s 238ms/step - loss: 0.0048
- val_loss: 0.0592

Epoch 00029: val_loss did not improve from 0.05885

Epoch 00029: ReduceLROnPlateau reducing learning rate to 1.000000047497451
4e-05.
Epoch 30/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0046
- val_loss: 0.0592

Epoch 00030: val_loss did not improve from 0.05885
Epoch 31/35
632/632 [==============================] - 151s 239ms/step - loss: 0.0045
- val_loss: 0.0592

Epoch 00031: val_loss did not improve from 0.05885
Epoch 00031: early stopping
```

```
Out[52]:

<tensorflow.python.keras.callbacks.History at 0x7fcdb75fbd68>
```

In [53]:

```
final_output = pred_Encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_siz
e, lstm_size, ita_txt_len, max_len_eng, 'general', att_units)
final_output.compile(optimizer = 'Adam', loss = loss_function)
final_output.load_weights('/content/ita_eng_3')
```

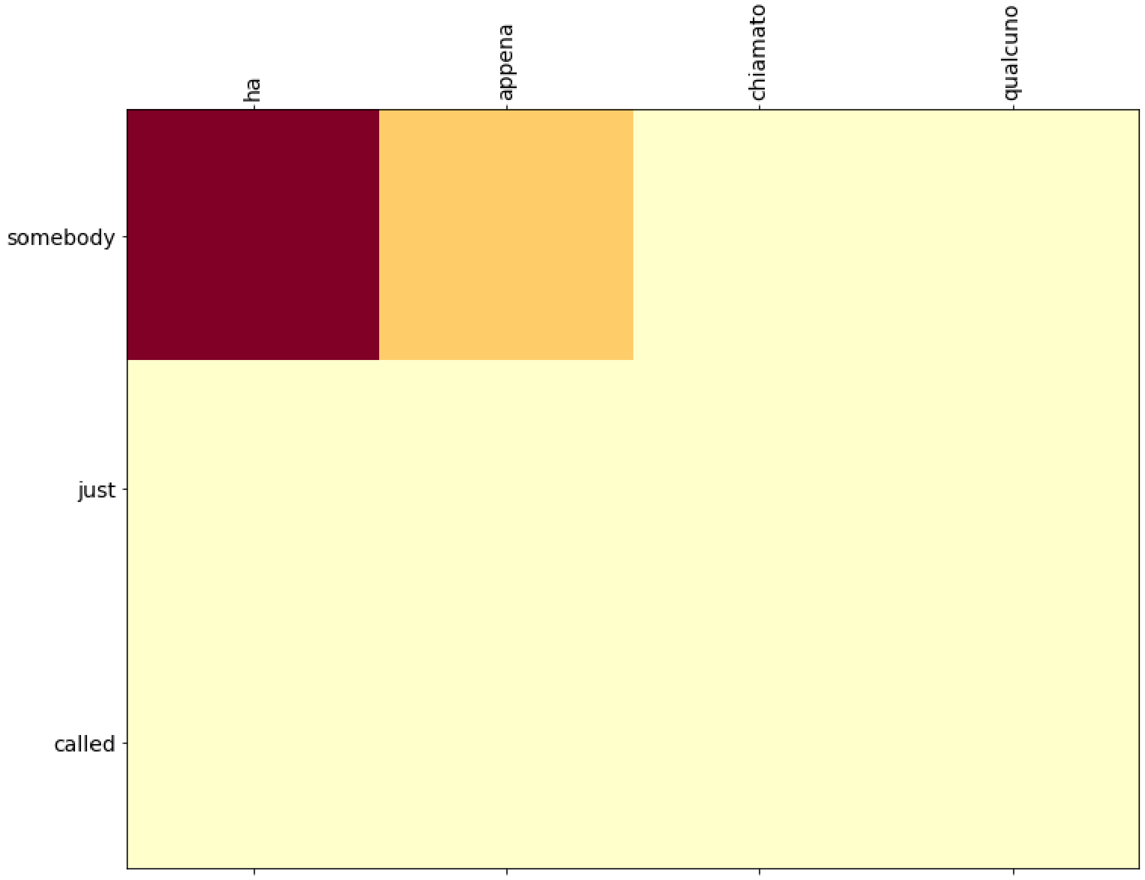
```
Out[53]:

<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcdb7
61c198>
```

In [54]:

```
pr_final = train['italin'].values[0][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[0])
```

```
input :  ha appena chiamato qualcuno
```

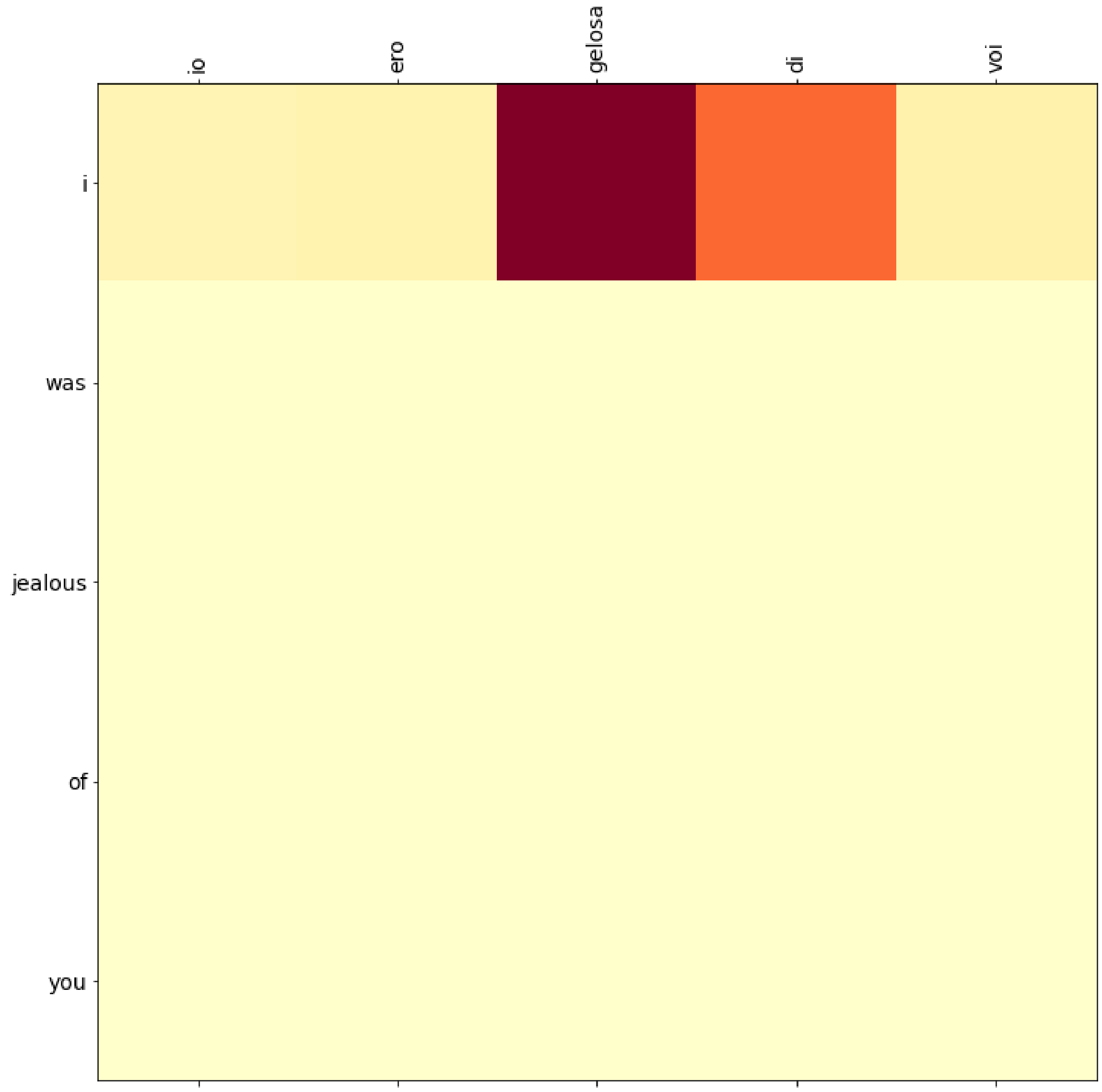


```
output :  somebody just called
```

In [55]:

```
pr_final = train['italin'].values[1000][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[1000])
```

```
input :  io ero gelosa di voi
```



```
output :  i was jealous of you
```

In [56]:

```
from nltk.translate.bleu_score import sentence_bleu
initial =0
for i in range(1000):
    pr_final = test['italin'].values[i][6:-6]
    con = test['english'].values[i]
    pred = predict(pr_final)
    initial+= sentence_bleu([con.split()], pred.split())
print('Bleu Score : {}'.format(score/1000))
```

```
Bleu Score : 0.8442320971046736
```

Repeat the same steps for Concat scoring function

In [57]:

```
from tensorflow.keras.callbacks import*
import os
batch_size=64
lstm_size=128
embedding_dim = 100

tr_dat = Dataset(train, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
te_dat  = Dataset(test, tokenizer_italian, tokenizer_english, ita_txt_len, max_len_eng)
val_dat  = Dataset(validation, tokenizer_italian, tokenizer_english, ita_txt_len, max_l
en_eng)

train_dataloader = Dataloder(tr_dat, batch_size=batch_size)
test_dataloader = Dataloder(te_dat, batch_size=batch_size)
val_dataloader = Dataloder(val_dat, batch_size = batch_size)
```

In [58]:

```
model = encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_size, lstm_size
, ita_txt_len, max_len_eng, 'concat', att_units, batch_size)
model.compile(optimizer = 'Adam', loss = loss_function)
log_dir="logs/seq2seq/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S") call_back =
[ModelCheckpoint('ita_eng_4', save_best_only= True),
             TensorBoard(log_dir = log_dir, histogram_freq=1, write_graph=True), EarlyS
topping(patience = 5, verbose = 1),
             ReduceLROnPlateau(patience = 3, verbose = 1)]
```

In [59]:

```
model.fit(x = train_dataloader, steps_per_epoch = train_dataloader.__len__(), validation_data = val_dataloader,
          validation_steps = val_dataloader.__len__(), epochs = 35, verbose = 1, callbacks = call_back)
```

```
Epoch 1/35
1265/1265 [==============================] - 319s 212ms/step - loss: 0.496
6 - val_loss: 0.3452
Epoch 2/35
1265/1265 [==============================] - 248s 196ms/step - loss: 0.322
5 - val_loss: 0.2693
Epoch 3/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.252
4 - val_loss: 0.2244
Epoch 4/35
1265/1265 [==============================] - 243s 192ms/step - loss: 0.206
7 - val_loss: 0.1907
Epoch 5/35
1265/1265 [==============================] - 243s 192ms/step - loss: 0.170
4 - val_loss: 0.1637
Epoch 6/35
1265/1265 [==============================] - 240s 190ms/step - loss: 0.138
3 - val_loss: 0.1394
Epoch 7/35
1265/1265 [==============================] - 238s 188ms/step - loss: 0.110
7 - val_loss: 0.1197
Epoch 8/35
1265/1265 [==============================] - 239s 189ms/step - loss: 0.087
3 - val_loss: 0.1049
Epoch 9/35
1265/1265 [==============================] - 242s 192ms/step - loss: 0.068
9 - val_loss: 0.0935
Epoch 10/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.055
0 - val_loss: 0.0849
Epoch 11/35
1265/1265 [==============================] - 244s 193ms/step - loss: 0.044
6 - val_loss: 0.0794
Epoch 12/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.036
8 - val_loss: 0.0747
Epoch 13/35
1265/1265 [==============================] - 248s 196ms/step - loss: 0.030
8 - val_loss: 0.0713
Epoch 14/35
1265/1265 [==============================] - 246s 195ms/step - loss: 0.026
0 - val_loss: 0.0692
Epoch 15/35
1265/1265 [==============================] - 245s 193ms/step - loss: 0.022
5 - val_loss: 0.0685
Epoch 16/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.020
0 - val_loss: 0.0666
Epoch 17/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.017
4 - val_loss: 0.0656
Epoch 18/35
1265/1265 [==============================] - 246s 194ms/step - loss: 0.015
6 - val_loss: 0.0645
Epoch 19/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.014
1 - val_loss: 0.0652
Epoch 20/35
1265/1265 [==============================] - 246s 194ms/step - loss: 0.013
1 - val_loss: 0.0638
Epoch 21/35
```

```
1265/1265 [==============================] - 246s 195ms/step - loss: 0.012
1 - val_loss: 0.0649
Epoch 22/35
1265/1265 [==============================] - 248s 196ms/step - loss: 0.011
0 - val_loss: 0.0648
Epoch 23/35
1265/1265 [==============================] - 246s 195ms/step - loss: 0.010
4 - val_loss: 0.0641

Epoch 00023: ReduceLROnPlateau reducing learning rate to 0.000100000004749
74513.
Epoch 24/35
1265/1265 [==============================] - 245s 194ms/step - loss: 0.008
3 - val_loss: 0.0622
Epoch 25/35
1265/1265 [==============================] - 248s 196ms/step - loss: 0.006
9 - val_loss: 0.0620
Epoch 26/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.006
4 - val_loss: 0.0623
Epoch 27/35
1265/1265 [==============================] - 248s 196ms/step - loss: 0.006
2 - val_loss: 0.0622
Epoch 28/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.006
0 - val_loss: 0.0627

Epoch 00028: ReduceLROnPlateau reducing learning rate to 1.000000047497451
4e-05.
Epoch 29/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.005
7 - val_loss: 0.0626
Epoch 30/35
1265/1265 [==============================] - 247s 195ms/step - loss: 0.005
7 - val_loss: 0.0626
Epoch 00030: early stopping
```

Out[59]:

```
<tensorflow.python.keras.callbacks.History at 0x7fcda3e93438>
```

In [60]:

```
final_output = pred_Encoder_decoder(set_ita_size, set_eng_size, embedding_dim, lstm_siz
e, lstm_size, ita_txt_len, max_len_eng, 'concat', att_units)
final_output.compile(optimizer = 'Adam', loss = loss_function)
final_output.load_weights('/content/ita_eng_4')
```

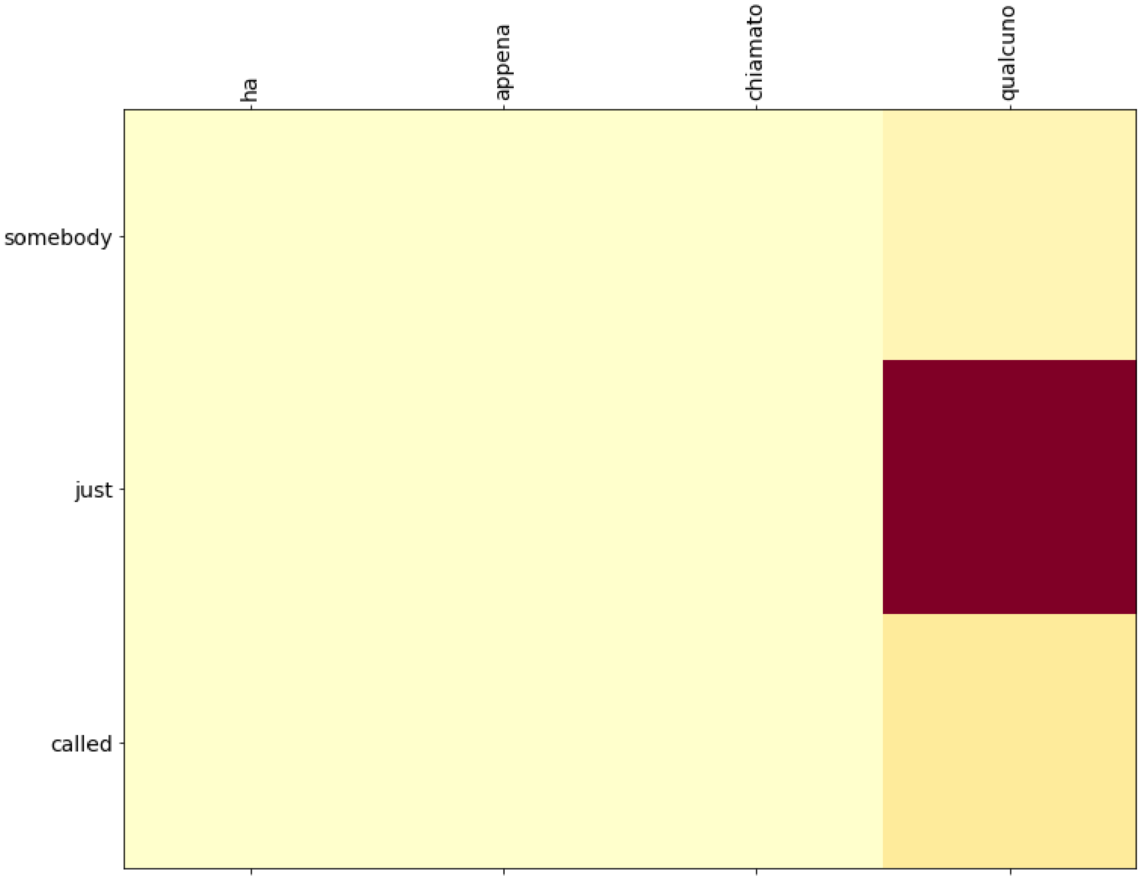
Out[60]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7fcd49
ac8c18>
```

In [61]:

```
pr_final = train['italin'].values[0][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[0])
```

```
input :  ha appena chiamato qualcuno
```

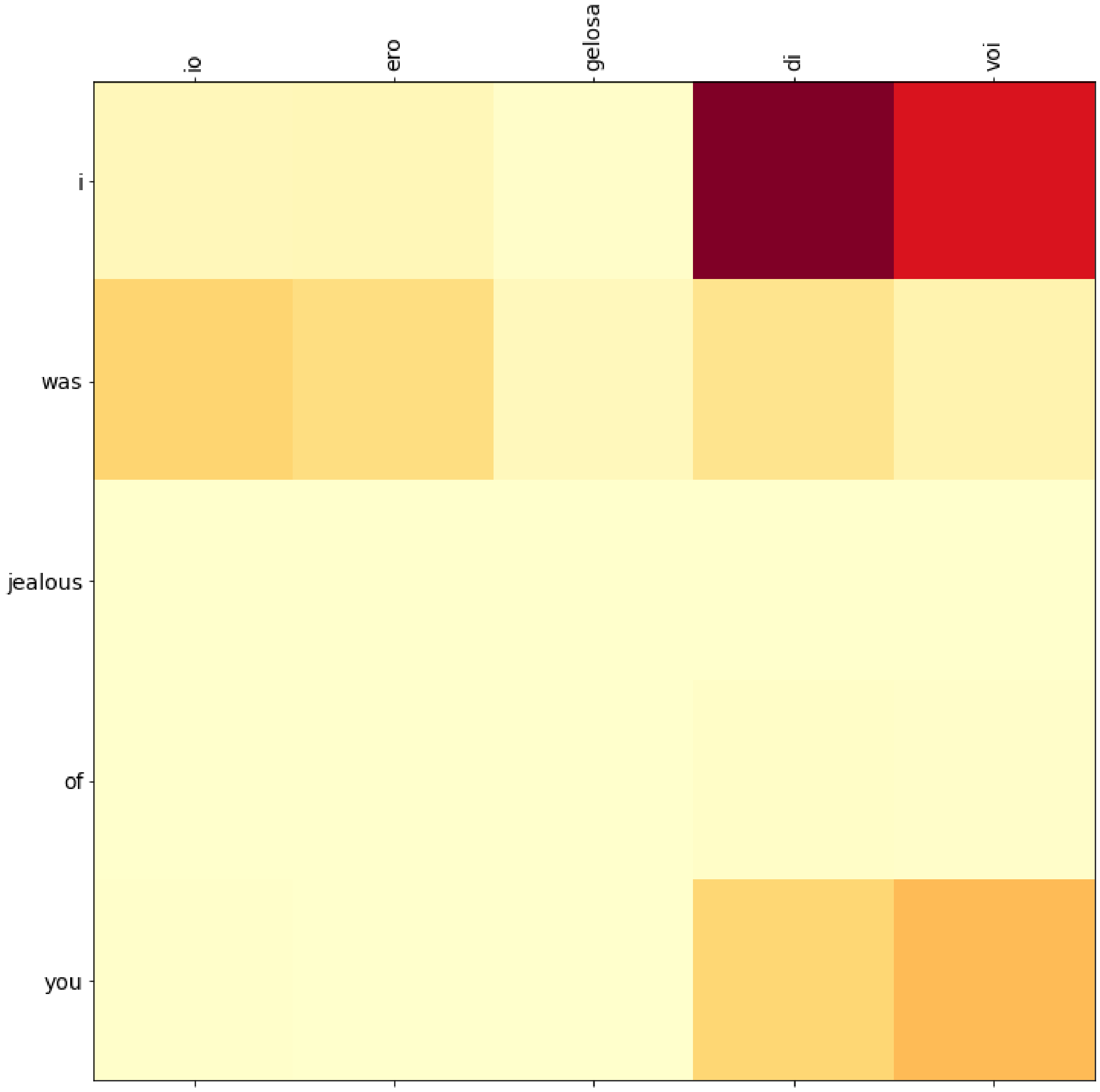


```
output :  somebody just called
```

In [62]:

```
pr_final = train['italin'].values[1000][6:-6]
print('input : ', pr_final)
result = predict(pr_final)
print('predicted output : ',result)
print('actual output :', train['english'].values[1000])
```

```
input :  io ero gelosa di voi
```



```
output :  i was jealous of you
```

In [63]:

```
from nltk.translate.bleu_score import sentence_bleu
initial =0
for i in range(1000):
    pr_final = test['italin'].values[i][6:-6]
    con = test['english'].values[i]
    pred = predict(pr_final)
    initial+= sentence_bleu([con.split()], pred.split())
print('Bleu Score : {}'.format(score/1000))
```

```
Bleu Score : 0.8438060676707472
```

Conclusion

- Bleu Score for model-1 is 0.8391122289889017
- Bleu Score for model-2 is 0.8262435337297614 (Attention mechanisum)
- Bleu Score for model-3 is 0.8442320971046736 (General scoring function)
- Bleu Score for model-4 is 0.8438060676707472 (Concat scoring function)