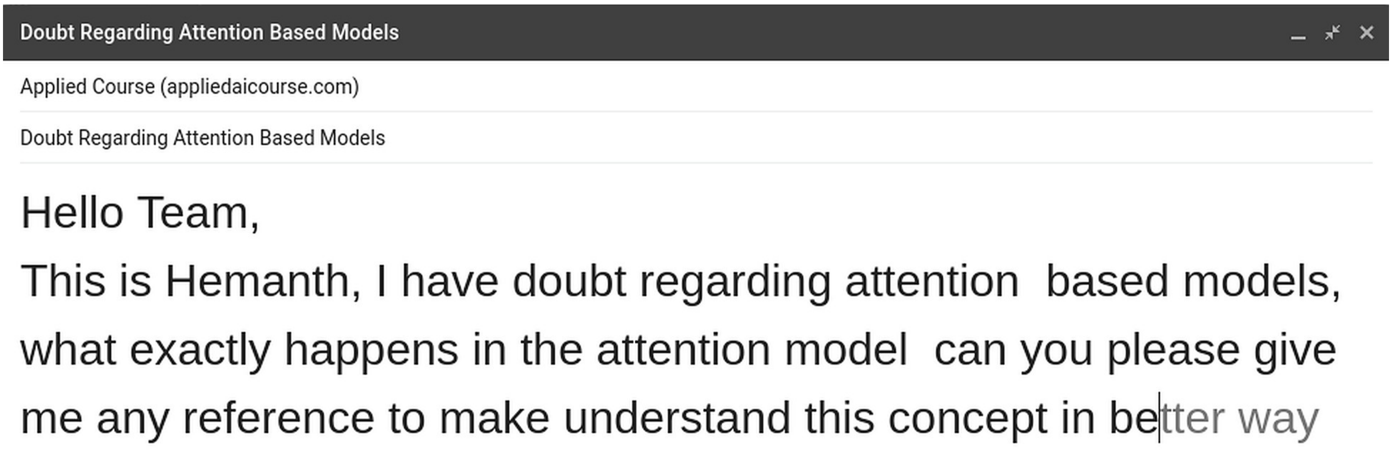
# Google Smart Composer

Google Smart Composer is a machine learning model that generates interactive word(s) suggestions. Given to whom you are writing the email, subject of the email previous email content(if a replied email) and a small part of the content of the email then the next coming word(s) is predicted by the model



Doubt Regarding Attention Based Models

Applied Course (appliedaicourse.com)

Doubt Regarding Attention Based Models

Hello Team,
This is Hemanth, I have doubt regarding attention  based models, what exactly happens in the attention model  can you please give me any reference to make understand this concept in better way

## Data:

Data is taken from personal emails and constructed according to problem specific.

### Structure of data:

From all the emails 'to', 'subject', 'previous email'(if any) and 'content' parts are taken. As we are going to predict next comming word in the content part, content part is breaked into following way. We are restriction our self to predict only atmost 5 next comming words so each email content will be breaked to many sentances.
**Ex:** Content: This is introduction to my project

| Sentance | Output |
|---:|---:|
| This | is |
| This | is introduction |
| This | is introduction to |
| This | is introduction to my |
| This | is introduction to my project |
| This is | introduction |
| This is | introduction to |
| This is | introduction to my |
| This is | introduction to my |
| This is | introduction to my project |
| This is introduction | to |
| This is introduction | to my |
| This is introduction | to my project |

Now to the sentance part 'to','subject', 'previous email' parts are joined with their corresponding separaters.
**Ex:** < to > email@ email.com< sub > introduction < prv > nan < cont > hello email this is only an intro

## Importing Libriries

In [ ]:

```
from __future__ import absolute_import, division, print_function, unicode_literals

try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass
import tensorflow as tf

import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
from sklearn.model_selection import train_test_split
import pandas as pd



import re
import numpy as np
import os
import io
import time
```

In [2]:

```
from google.colab import drive
drive.mount('gdrive',force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/aut
h?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleu
sercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&respon
se_type=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fd
ocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%
2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2
fwww.googleapis.com%2fauth%2fpeopleapi.readonly (https://accounts.goog
le.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc
0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoaut
h%3a2.0%3aoob&response_type=code&scope=email%20https%3a%2f%2fwww.googl
eapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2faut
h%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.rea
donly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly)

Enter your authorization code:
··········
Mounted at gdrive
```

## Loading Data

In [3]:

```
data = pd.read_csv('gdrive/My Drive/google/final_data_my_emails.csv')
print('Number of rows in data',data.shape[0])
print('Number of columns in data',data.shape[1])
data.head()
```

```
Number of rows in data 37230
Number of columns in data 2
```

Out[3]:

| | x | y |
|---|---|---|
| **0** | <to> yernagulahemanth <prv> nan <sub> about f... | yernagulahemanth |
| **1** | <to> yernagulahemanth <prv> nan <sub> about f... | yernagulahemanth can |
| **2** | <to> yernagulahemanth <prv> nan <sub> about f... | yernagulahemanth can you |
| **3** | <to> yernagulahemanth <prv> nan <sub> about f... | yernagulahemanth can you explain |
| **4** | <to> yernagulahemanth <prv> nan <sub> about f... | yernagulahemanth can you explain more |

In [4]:

```
data.y = data.y.apply(lambda x:str(x)) # y part was having now int values so they a

# Along with defalt tags x and y are added with <start> and <end> tags at starting
data.x = data.x.apply(lambda x:'<start> ' + str(x) + ' <end>')
data.y = data.y.apply(lambda x:'<start> ' + str(x) + ' <end>')

data.head()
```

Out[4]:

| | x | y |
|---|---|---|
| **0** | <start> <to> yernagulahemanth <prv> nan <sub> ... | <start> yernagulahemanth <end> |
| **1** | <start> <to> yernagulahemanth <prv> nan <sub> ... | <start> yernagulahemanth can <end> |
| **2** | <start> <to> yernagulahemanth <prv> nan <sub> ... | <start> yernagulahemanth can you <end> |
| **3** | <start> <to> yernagulahemanth <prv> nan <sub> ... | <start> yernagulahemanth can you explain <end> |
| **4** | <start> <to> yernagulahemanth <prv> nan <sub> ... | <start> yernagulahemanth can you explain more ... |

In [0]:

```
# Tokenizing x and y
# Tokenizing: Collecting all tokens(words) for x and y and storing them  with index

X_  = list(data.x.values)
Y_ = list(data.y.values)



# Tokenizing x values, since default values of filters in tokenizer is special char
# default value by ''
x_tokenizer   = tf.keras.preprocessing.text.Tokenizer(filters='')
x_tokenizer.fit_on_texts(X_)
x_tensor         = x_tokenizer.texts_to_sequences(X_)

# padding
x_tensor         = tf.keras.preprocessing.sequence.pad_sequences(x_tensor,padding='


y_tokenizer  = tf.keras.preprocessing.text.Tokenizer(filters='')
y_tokenizer.fit_on_texts(Y_)
y_tensor        = y_tokenizer.texts_to_sequences(Y_)
y_tensor        = tf.keras.preprocessing.sequence.pad_sequences(y_tensor,padding='p
```

In [0]:

```
# Y_
```

In [7]:

```
print('Total number of words in x:',len(x_tokenizer.index_word))
print('Total number of words in y:',len(y_tokenizer.index_word))
```

```
Total number of words in x: 1449
Total number of words in y: 1417
```

In [8]:

```
# Making train and validation data with 80-20 ratio
x_train, x_val, y_train, y_val = train_test_split(x_tensor, y_tensor, test_size=0.2

# Show length
y_token_max_len, x_token_max_len = max([len(i) for i in y_tensor]), max([len(i) for

print('Number of data points in xtrain:',len(x_train))
print('Number of data points in ytrain:',len(y_train))
print('Number of data points in xval  :',len(x_val))
print('Number of data points in yval  :',len(y_val))
```

```
Number of data points in xtrain: 29784
Number of data points in ytrain: 29784
Number of data points in xval  : 7446
Number of data points in yval  : 7446
```

In [0]:

```
y_token_max_len,x_token_max_len
```

In [0]:

```
```

## Feeding this data into tensorflow data api

we can make operation like shuffleing, getting batch wise so as to make future simple

`Ref: https://www.tensorflow.org/guide/data' (https://www.tensorflow.org/guide/data')

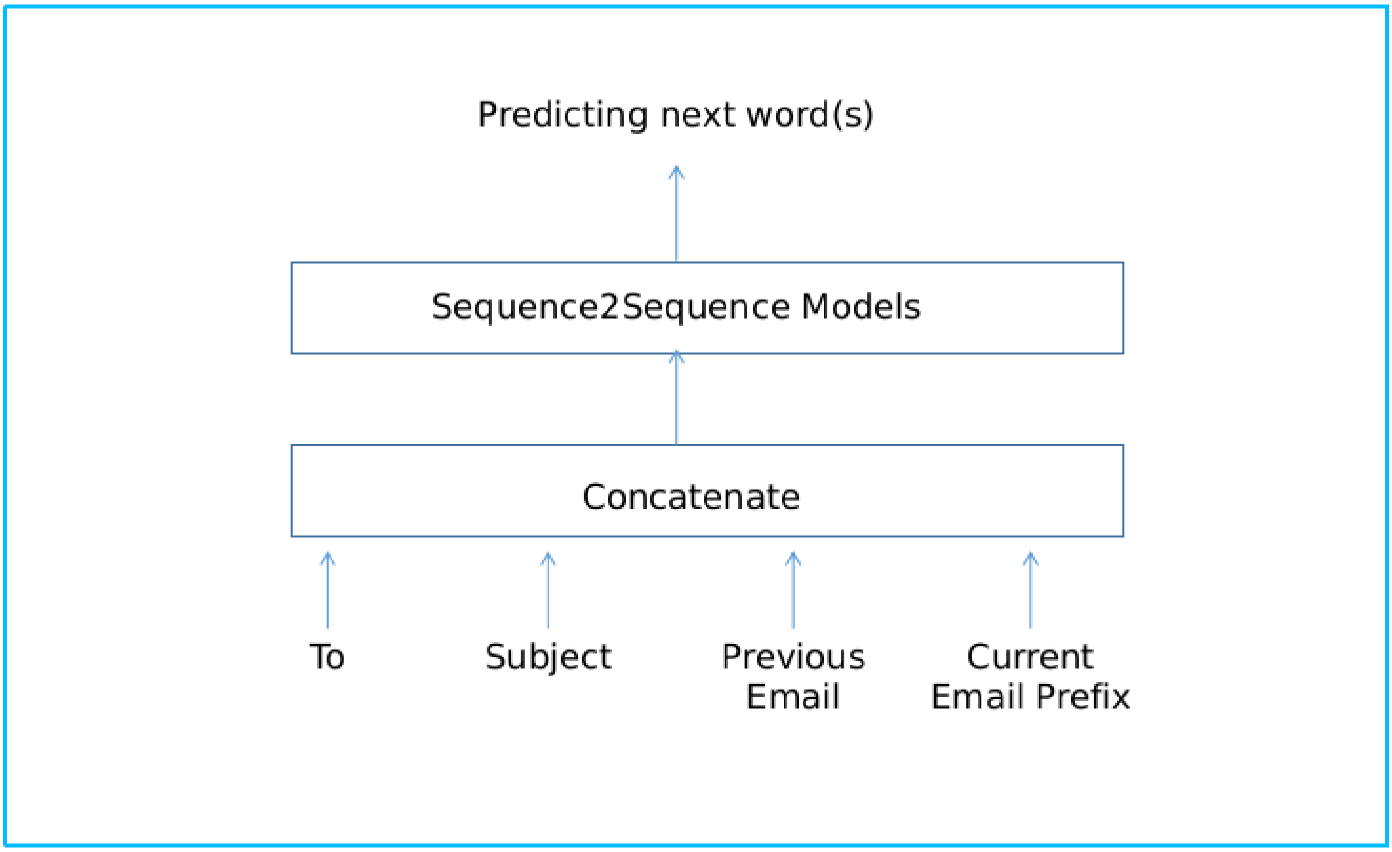
In [0]:

```
Batch_Size = 50
steps_per_epoch = len(x_train)//Batch_Size
Emd_dim = 200
units = 500
x_vocab_size = len(x_tokenizer.word_index)+1
y_vocab_size = len(y_tokenizer.word_index)+1

# Shuffling the data set
dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train)).shuffle(len(x_trai
# Getting batch wise and droping last batch if it is not having less number of poin
dataset = dataset.batch(Batch_Size, drop_remainder=True)
```

In [0]:

```
```

## Encoder Decoder With ATTENTION

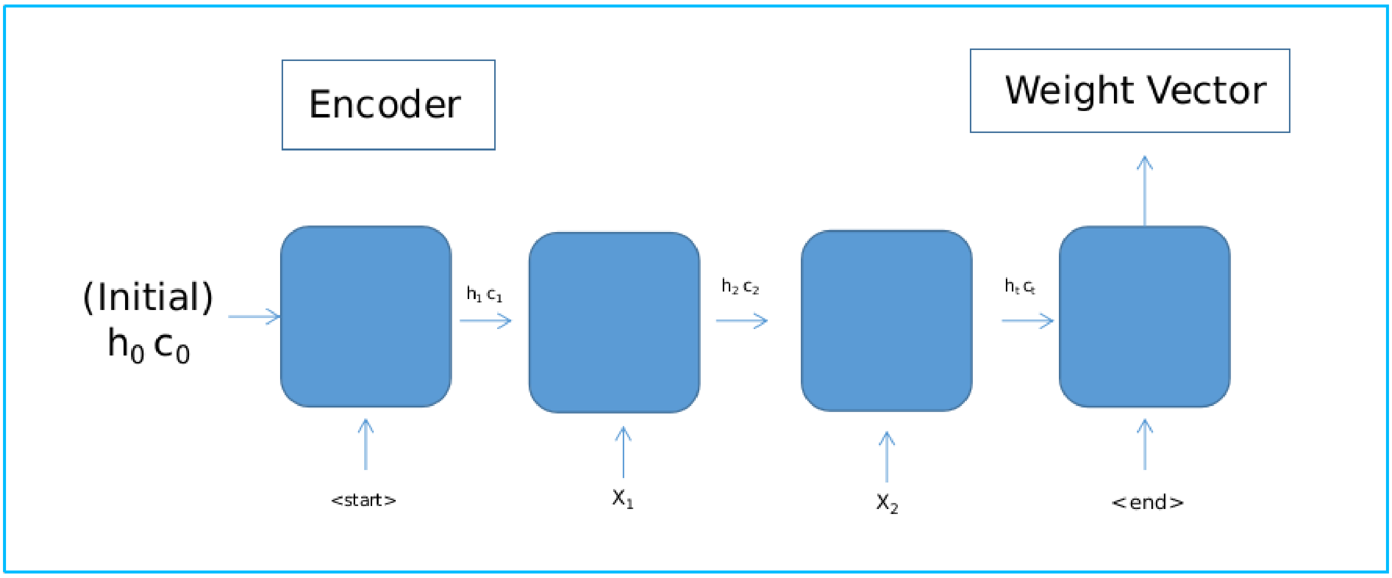


**Short Explination:**

1. The 'to', 'subject', 'previous email', 'current email prefix' is concatenated.
2. This sequence is sent into sequence model in our case encoder, decoder with attention
3. This model predicts the next comming words after current email prefix

## Encoder Layer

Every time when a sentance is passed to the encoder, at every time step the outputs are depricated and hidden states from every time steps pass to next time step at the end it generates output and a weight vector which hold whole context of the sentance



In [0]:

```
class Encoder(tf.keras.Model):

  def __init__(self, vocab_size, Emd_dim, enc_units, batch_size):
    '''
    vocab_size   - Vocabulary size
    Emd_dim      - Number of emding dimensions
    batch_size   - Batch Size

    '''
    super(Encoder, self).__init__()
    self.batch_size = batch_size
    self.enc_units = enc_units
    self.embedding = tf.keras.layers.Embedding(vocab_size, Emd_dim)
    self.encoder = tf.keras.layers.LSTM(self.enc_units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')

  def call(self, input_, hidden):

    '''
      Given input and hidden states for encoder returns the updated hidden states a
    '''
    input_ = self.embedding(input_)
    # print(x.shape,hidden[0].shape,hidden[1].shape)
    output, state_h,state_c = self.encoder(input_, initial_state = hidden)
    # state = tf.concat([state_h,state_c],axis=1)
    state = [state_h,state_c]
    return output, state


  def initialize_hidden_state(self):
      '''
       Returns initial states for encoder (zeros)
      '''
      initial_states = [tf.zeros((self.batch_size, self.enc_units)),tf.zeros((self.
      return initial_states
```

In [0]:

```
encoder = Encoder(x_vocab_size, Emd_dim, units, Batch_Size)
```

## Attention Layer

In each time step the decoder hidden state and encoder outputs are passed through dense layer so that they get their weights and can be trained through back propagation, both this weights passed from a dense layer is passed through tanh activation function and than passed through dense unit with single activation unit upon which softmax activation is applied so that sum of the vector will be sum to one which is known as attention weights

Now this attention weights are element wise multiplied encoder output which produce context vector, this context vector will be holding all important words with high values and less important words with low values as we have applied softmax activation function

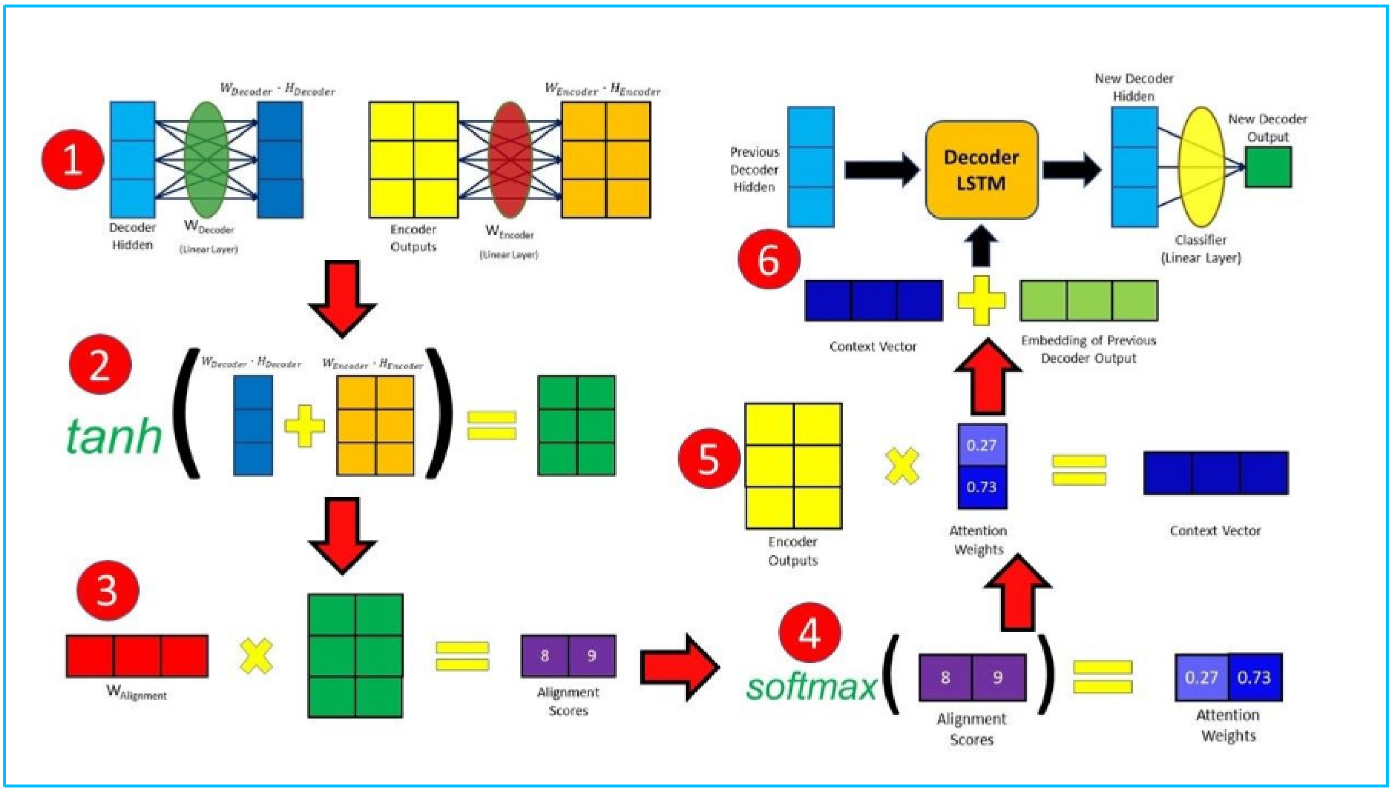


***Image courtesy : https://blog.floydhub.com/attention-mechanism/.***

In [0]:

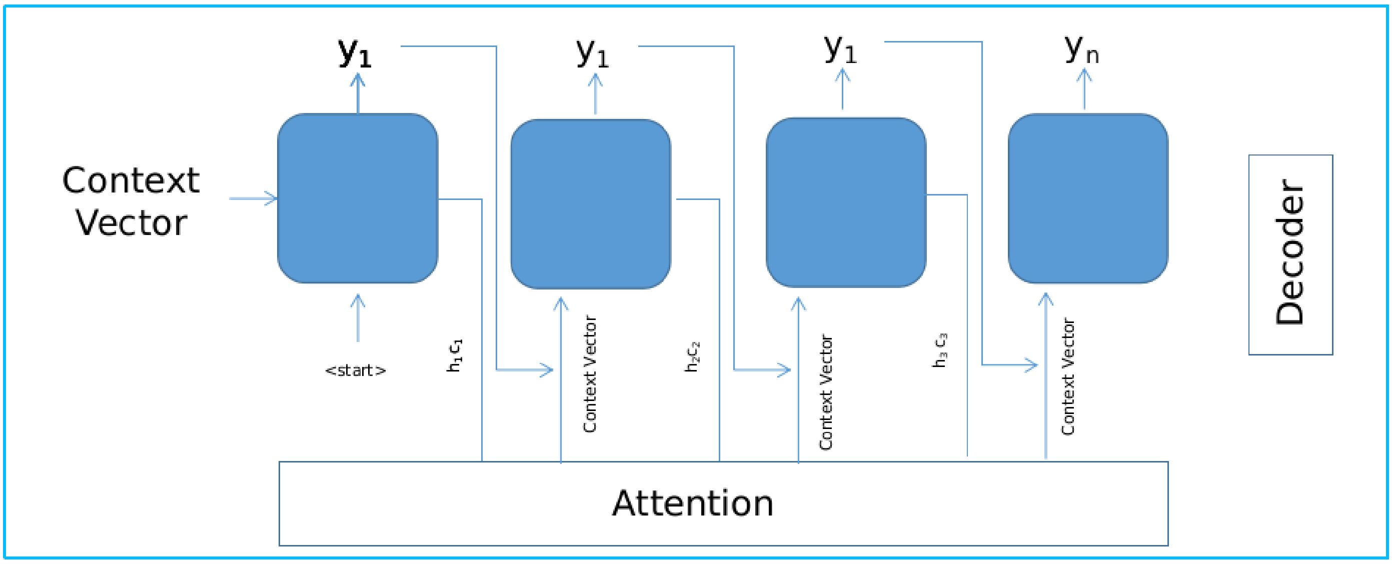
```
class Attention(tf.keras.layers.Layer):
  def __init__(self, units):
    super(Attention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)
    self.units = units

  def call(self, hid, enc_out):

    # hidden shape == (Batch_Size, hidden size)
    # hidden_with_time_axis shape == (Batch_Size, 1, hidden size)
    # we are doing this to perform addition to calculate the score
    # hidden_with_time_axis = tf.expand_dims(hid, 1)
    # Since Lstm layers will give us two states i.e cell state and hidden state
    # we are passing hidden state through dense layer and cell state through dence
    # and adding both outputs of dence  layer
    hidden_with_time_axis  = tf.expand_dims(self.W2(hid[0])+self.W2(hid[1]),1)
    # print(hidden_with_time_axis.shape)
    # Now this sum of  outputs from dense layers is added with outputs of  encoder
    # along with tanh and a single unit dense layer
    # score shape == (Batch_Size, max_length, 1)
    # we get 1 at the last axis because we are applying score to self.V
    # the shape of the tensor before applying self.V is (batch_size, max_length, un
    score = self.V(tf.nn.tanh(
        self.W1(enc_out) + hidden_with_time_axis))

    # attention_weights shape == (batch_size, max_length, 1)
    attention_weights = tf.nn.softmax(score, axis=1)

    # context_vector shape after sum == (batch_size, hidden_size)
    context_vector = attention_weights * enc_out
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weights
```

In [0]:

```
attention_layer = Attention(10)
```

## Decoder

For the first time step the context vector generated with encoder output and encoder hidden states are passed to the decoder initial stage. From next time step onwards the output producted from decoder in previous time step and hidden states for decoder in previous time step are passed to attention layer which generates context vector, then this context vector concatnated with decoder input and passed into the decoder layer, this process happens till the end tag (< end >) reached in the sequence


y1
y1
y1
yn
Context Vector
Decoder
<start>
h1c1
Context Vector
h2c2
Context Vector
h3 c3
Context Vector
Attention

In [0]:

```
class Decoder(tf.keras.Model):
  def __init__(self, vocab_size, Emd_dim, dec_units, batch_size,nwtpa = 10):
    '''
    All the variables are same as previous encoder calss except nwtpa
    nwtpa is number of units you want to maintain at attention mechanism
    '''

    super(Decoder, self).__init__()
    self.batch_size = batch_size
    self.dec_units = dec_units
    self.embedding = tf.keras.layers.Embedding(vocab_size, Emd_dim)
    self.decoder = tf.keras.layers.LSTM(self.dec_units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')
    self.fc = tf.keras.layers.Dense(vocab_size)
    self.nwtpa = nwtpa

    # used for attention
    self.attention = Attention(self.nwtpa)

  def call(self, input_, hidden, enc_output):
    # enc_output shape == (batch_size, max_length, hidden_size)
    # passing  hidden states and encoder output to the attention layer
    context_vector, attention_weights = self.attention(hidden, enc_output)
    # applying embdding to the inputs of the dense layer
    # input_ shape after passing through embedding == (batch_size, 1, Emd_dim)
    input_ = self.embedding(input_)

    # concating previous input and context vector
    # input_ shape after concatenation == (batch_size, 1, Emd_dim + hidden_size)
    input_ = tf.concat([tf.expand_dims(context_vector, 1), input_], axis=-1)

    # passing the concatenated vector to the Lstm
    output, state_h,state_c = self.decoder(input_)
    state = [state_h,state_c]

    # output shape == (batch_size * 1, hidden_size)
    output = tf.reshape(output, (-1, output.shape[2]))

    # output shape == (batch_size, vocab)
    output = self.fc(output)

    return output, state, attention_weights
```

In [0]:

```
decoder = Decoder(y_vocab_size, Emd_dim, units, Batch_Size)
```

## Define the optimizer and the loss function

In [0]:

```
optimizer = tf.keras.optimizers.Adam()
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

def loss_function(actual, pred):
  mask = tf.math.logical_not(tf.math.equal(actual, 0))
  loss_ = loss_object(actual, pred)

  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask

  return tf.reduce_mean(loss_)
```

## Creating Checkpoints

In [0]:

```
checkpoint_dir = './checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(optimizer=optimizer, encoder=encoder, decoder=deco
```

## Training

In [0]:

```
# for i in os.listdir('training_checkpoints'):
#     os.remove('training_checkpoints/'+i)
```

In [0]:

```
def train(EPOCHS = 10,verbose = 0):

      '''
          Given number of epochs and verbose model is trainied with those number of
          verbose - If verbose is 0 no information about batches is prednted, if ve
                    everytime when batch reaches batch_size/verbose info is printed
      '''
      for epoch in range(EPOCHS):
            start = time.time()

            # initial hidden states for encoder is being generated
            enc_hidden = encoder.initialize_hidden_state()
            total_loss = 0

            for (batch, (inp, targ)) in enumerate(dataset.take(steps_per_epoch)):
                    # batch_loss = train_step(inp, targ, enc_hidden)
                    # total_loss += batch_loss
                    loss = 0

                    with tf.GradientTape() as tape:

                          # encoder hidden state and x(inp) is passed to encoder wh
                          # encoder output and its hidden states
                          enc_output, enc_hidden = encoder(inp, enc_hidden)

                          dec_hidden = enc_hidden

                          # for the first time decoder input will be only <start> t

                          dec_input = tf.expand_dims([y_tokenizer.word_index['<star
                          for t in range(1, targ.shape[1]):
                              # passing enc_output, dec_input, dec_hidden state to
                              predictions, dec_hidden, _ = decoder(dec_input, dec_h

                              loss += loss_function(targ[:, t], predictions)

                              # using teacher forcing
                              dec_input = tf.expand_dims(targ[:, t], 1)
                    batch_loss = (loss / int(targ.shape[1]))

                    variables = encoder.trainable_variables + decoder.trainable_var

                    gradients = tape.gradient(loss, variables)

                    optimizer.apply_gradients(zip(gradients, variables))

                    if verbose:
                        if batch % verbose == 0:
                              print('Epoch {} Batch {} Loss {}'.format(epoch + 1, b

            # saving (checkpoint) the model every 2 epochs
            if (epoch + 1) % 2 == 0:
              checkpoint.save(file_prefix = checkpoint_prefix)

            print('Epoch {} Loss {}'.format(epoch + 1, total_loss / steps_per_epoch
            print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```

In [20]:

```
train(verbose=100)
```

```
Epoch 1 Batch 0 Loss 4.1264
Epoch 1 Batch 100 Loss 2.9349
Epoch 1 Batch 200 Loss 2.7352
Epoch 1 Batch 300 Loss 3.0306
Epoch 1 Batch 400 Loss 3.0160
Epoch 1 Batch 500 Loss 2.6029
Epoch 1 Loss 0.0
Time taken for 1 epoch 127.41328072547913 sec

Epoch 2 Batch 0 Loss 2.1338
Epoch 2 Batch 100 Loss 2.1332
Epoch 2 Batch 200 Loss 1.9065
Epoch 2 Batch 300 Loss 1.9896
Epoch 2 Batch 400 Loss 1.5953
Epoch 2 Batch 500 Loss 1.6653
Epoch 2 Loss 0.0
Time taken for 1 epoch 112.77655529975891 sec

Epoch 3 Batch 0 Loss 1.5814
Epoch 3 Batch 100 Loss 1.5359
Epoch 3 Batch 200 Loss 1.5008
Epoch 3 Batch 300 Loss 1.4783
Epoch 3 Batch 400 Loss 1.4984
Epoch 3 Batch 500 Loss 1.3349
Epoch 3 Loss 0.0
Time taken for 1 epoch 112.18088150024414 sec

Epoch 4 Batch 0 Loss 1.2326
Epoch 4 Batch 100 Loss 1.2452
Epoch 4 Batch 200 Loss 1.1833
Epoch 4 Batch 300 Loss 1.2856
Epoch 4 Batch 400 Loss 1.2822
Epoch 4 Batch 500 Loss 1.1809
Epoch 4 Loss 0.0
Time taken for 1 epoch 114.6331856250763 sec

Epoch 5 Batch 0 Loss 1.2219
Epoch 5 Batch 100 Loss 1.1174
Epoch 5 Batch 200 Loss 1.0799
Epoch 5 Batch 300 Loss 1.1781
Epoch 5 Batch 400 Loss 1.0822
Epoch 5 Batch 500 Loss 0.9566
Epoch 5 Loss 0.0
Time taken for 1 epoch 113.33757972717285 sec

Epoch 6 Batch 0 Loss 0.9424
Epoch 6 Batch 100 Loss 0.9333
Epoch 6 Batch 200 Loss 0.9851
Epoch 6 Batch 300 Loss 0.9313
Epoch 6 Batch 400 Loss 0.9915
Epoch 6 Batch 500 Loss 0.9855
Epoch 6 Loss 0.0
Time taken for 1 epoch 113.54746222496033 sec

Epoch 7 Batch 0 Loss 0.8394
Epoch 7 Batch 100 Loss 0.9163
Epoch 7 Batch 200 Loss 0.8759
Epoch 7 Batch 300 Loss 0.8819
Epoch 7 Batch 400 Loss 0.8631
Epoch 7 Batch 500 Loss 0.8246
Epoch 7 Loss 0.0
Time taken for 1 epoch 112.03907346725464 sec

Epoch 8 Batch 0 Loss 0.8177
Epoch 8 Batch 100 Loss 0.7740
Epoch 8 Batch 200 Loss 0.7412
Epoch 8 Batch 300 Loss 0.8472
Epoch 8 Batch 400 Loss 0.8170
Epoch 8 Batch 500 Loss 0.8224
Epoch 8 Loss 0.0
Time taken for 1 epoch 111.9318687915802 sec

Epoch 9 Batch 0 Loss 0.7100
Epoch 9 Batch 100 Loss 0.8428
Epoch 9 Batch 200 Loss 0.7434
Epoch 9 Batch 300 Loss 0.7513
Epoch 9 Batch 400 Loss 0.8366
Epoch 9 Batch 500 Loss 0.8006
Epoch 9 Loss 0.0
Time taken for 1 epoch 111.95030188560486 sec

Epoch 10 Batch 0 Loss 0.6798
Epoch 10 Batch 100 Loss 0.7494
Epoch 10 Batch 200 Loss 0.7031
Epoch 10 Batch 300 Loss 0.6558
Epoch 10 Batch 400 Loss 0.6665
Epoch 10 Batch 500 Loss 0.6705
Epoch 10 Loss 0.0
Time taken for 1 epoch 112.42571496963501 sec
```

In [0]:

```
def test(sentence):
  attention_plot = np.zeros((y_token_max_len, x_token_max_len))

  # Tokenizing inputs
  inputs = [x_tokenizer.word_index[i] for i in sentence.split()]
  # Padding inputs
  inputs = tf.keras.preprocessing.sequence.pad_sequences([inputs],
                                                         maxlen=x_token_max_len,
                                                         padding='post')
  inputs = tf.convert_to_tensor(inputs)

  result = ''

  # Making initial hidden states for encoder
  hidden = [tf.zeros((1, units)),tf.zeros((1, units))]

  # inputs and hidden states are passed to encoder
  enc_out, enc_hidden = encoder(inputs, hidden)

  dec_hidden = enc_hidden

  # Making initial inputs for decoder as <start> tag
  dec_input = tf.expand_dims([y_tokenizer.word_index['<start>']], 0)

  for t in range(y_token_max_len):
          # inputing decoder input(<start> during initial sate), encoder hidden sta
          predictions, dec_hidden,_ = decoder(dec_input, dec_hidden,enc_out)

          predicted_id = tf.argmax(predictions[0]).numpy()

          # the predicted word is appended to result

          result += y_tokenizer.index_word[predicted_id] + ' '

          # if the predicted word is <end> returning the words predicted till now
          if y_tokenizer.index_word[predicted_id] == '<end>':
            return result

          # current output from decoder is fed back to decoder into next timestep
          dec_input = tf.expand_dims([predicted_id], 0)

  return result
```

In [0]:

In [0]:

```
def predict_next_word(sentence):
  result = test(sentence)
  return result
```

## Restore the latest checkpoint and test

In [33]:

```
# restoring the latest checkpoint in checkpoint_dir
checkpoint_dir = 'gdrive/My Drive/google/check_points'
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
```

Out[33]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f
d1699abb38>
```

In [0]:

```
import shutil
# Moving checkpoints to drive
for i in os.listdir('training_checkpoints'):
  shutil.move('training_checkpoints/'+i,'gdrive/My Drive/google/check_points')
```

In [0]:

```
predict_next_word(u'<start> <to> team <prv> nan <sub> about mentor <cont> hello tea
```

## We shall check the model with some unseen data

In [0]:

```
def generate_sent(id_):

        test_with = []
        try:
          for i in x_val[id_]:
            test_with.append(x_tokenizer.index_word[i])
        except:

          return ' '.join(test_with)
```

In [26]:

```
input_ = generate_sent(5)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  classrooms one by one you will receive a communication  ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> team <prv> nan <sub> regarding new class room <cont
```

```
---------- input ----------
<start> <to> team <prv> nan <sub> regarding new class room <cont> hell
o team we are migrating classrooms one by one you will receive a commu
nication <end>


---------- actual ----------
  classrooms one by one you will receive a communication


---------- predicted ----------
```

Out[26]:

```
'classrooms one by one <end> '
```

In [30]:

```
input_ = generate_sent(128)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  netflix what s the first thing i should add to  ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> saitejapsk <prv> nan <sub> new netflix series <cont
```

```
---------- input ----------
<start> <to> saitejapsk <prv> nan <sub> new netflix series <cont> hell
o saitejapsk i m thinking of getting netflix what s the first thing i
should add to <end>


---------- actual ----------
  netflix what s the first thing i should add to


---------- predicted ----------
```

Out[30]:

```
'netflix what s the first <end> '
```

In [32]:

```
input_ = generate_sent(195)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('   finish my current projects in the sales department before i move  ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> hello yernagulahemanth wow t
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> hello yernagulahemanth wow this is
great news i am so glad for you so you will start your new job this co
ming monday on fri dec at yernagulahemanth <sub> good news <cont> hell
o yernagulahemanth casestudy no i need to finish my current projects i
n the sales department before i move <end>


---------- actual ----------
   finish my current projects in the sales department before i move


---------- predicted ----------
```

Out[32]:

```
'finish my current projects <end> '
```

In [34]:

```
input_ = generate_sent(184)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' learning proven track record of applying deep learning and machine learning
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <co
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <cont> he
llo yernagulahemanth placements about company gms role data science in
tern location mumbai interviews to ml interviews requirements very goo
d programming knowledge very good knowledge of ml and deep learning pr
oven track record of applying deep learning and machine learning note
preference would be given to people who <end>


---------- actual ----------
 learning proven track record of applying deep learning and machine le
arning note preference would be given to people who


---------- predicted ----------
```

Out[34]:

```
'learning proven track record of <end> '
```

In [35]:

```
input_ = generate_sent(184)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  deep learning and machine learning note preference would be given to peopl
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <co
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <cont> he
llo yernagulahemanth placements about company gms role data science in
tern location mumbai interviews to ml interviews requirements very goo
d programming knowledge very good knowledge of ml and deep learning pr
oven track record of applying deep learning and machine learning note
preference would be given to people who <end>


---------- actual ----------
  deep learning and machine learning note preference would be given to
people who


---------- predicted ----------
```

Out[35]:

```
'deep learning and machine <end> '
```

In [36]:

```
input_ = generate_sent(184)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  note preference would be given to people who  ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <co
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> opening in gms <cont> he
llo yernagulahemanth placements about company gms role data science in
tern location mumbai interviews to ml interviews requirements very goo
d programming knowledge very good knowledge of ml and deep learning pr
oven track record of applying deep learning and machine learning note
preference would be given to people who <end>


---------- actual ----------
  note preference would be given to people who


---------- predicted ----------
```

Out[36]:

```
'note preference would be given <end> '
```

In [40]:

```
input_ = generate_sent(7)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' resubmission reply if you dont want to receive emails from classroom you ca
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> 22assign <cont> he
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> 22assign <cont> hello ye
rnagulahemanth q ment rs applied ai course added a private comment on
assignment apply svm on donors choose dataset m applied ai course perf
ormance of your models are poor can you please try out any techniques
apart from the ones mentioned in the instructions and try to improve p
lease incorporate the above suggestions and resubmit your assignment a
nd also add a comment after resubmission reply if you dont want to rec
eive emails from classroom you can unsubscribe google inc <end>


---------- actual ----------
 resubmission reply if you dont want to receive emails from classroom
you can unsubscribe google inc


---------- predicted ----------
```

Out[40]:

```
'also add a comment <end> '
```

In [44]:

```
input_ = generate_sent(20)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('customer your coupons have been listed below if there is anything else we ca
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont> hell
o yernagulahemanth bookmyshow dear customer your coupons have been lis
ted below if there is anything else we can do to make you smile please
dont hesitate to get in touch with us off on pizzas on online ordering
only ovenstory pizza coupon code bmsosfifty validity may may t c impor
tant instructions the coupons listed above are completely owned by the
brands who are providing the offer bookmyshow will not be liable if th
e brands refuse to accept the coupons at any brand outlets please read
all terms conditions of the respective coupon the coupons listed above
can be used only once at a brand outlet any attempt <end>


---------- actual ----------
customer your coupons have been listed below if there is anything else
we can do to make you smile please dont hesitate to get in touch with
us off on pizzas on online ordering only ovenstory pizza coupon code b
msosfifty validity may may t c important instructions the coupons list
ed above are completely owned by the brands who are providing the offe
r bookmyshow will not be liable if the brands refuse to accept the cou
pons at any brand outlets please read all terms conditions of the resp
ective coupon the coupons listed above can be used only once at a bran
d outlet any attempt


---------- predicted ----------
```

Out[44]:

```
'customer your coupons have <end> '
```

In [45]:

```
input_ = generate_sent(20)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' if there is anything else we can do to make you smile please dont hesitate
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont> hell
o yernagulahemanth bookmyshow dear customer your coupons have been lis
ted below if there is anything else we can do to make you smile please
dont hesitate to get in touch with us off on pizzas on online ordering
only ovenstory pizza coupon code bmsosfifty validity may may t c impor
tant instructions the coupons listed above are completely owned by the
brands who are providing the offer bookmyshow will not be liable if th
e brands refuse to accept the coupons at any brand outlets please read
all terms conditions of the respective coupon the coupons listed above
can be used only once at a brand outlet any attempt <end>


---------- actual ----------
 if there is anything else we can do to make you smile please dont hes
itate to get in touch with us off on pizzas on online ordering only ov
enstory pizza coupon code bmsosfifty validity may may t c important in
structions the coupons listed above are completely owned by the brands
who are providing the offer bookmyshow will not be liable if the brand
s refuse to accept the coupons at any brand outlets please read all te
rms conditions of the respective coupon the coupons listed above can b
e used only once at a brand outlet any attempt


---------- predicted ----------
```

Out[45]:

```
'there is anything else <end> '
```

In [46]:

```
input_ = generate_sent(20)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  smile please dont hesitate to get in touch with us off on pizzas on online
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont> hell
o yernagulahemanth bookmyshow dear customer your coupons have been lis
ted below if there is anything else we can do to make you smile please
dont hesitate to get in touch with us off on pizzas on online ordering
only ovenstory pizza coupon code bmsosfifty validity may may t c impor
tant instructions the coupons listed above are completely owned by the
brands who are providing the offer bookmyshow will not be liable if th
e brands refuse to accept the coupons at any brand outlets please read
all terms conditions of the respective coupon the coupons listed above
can be used only once at a brand outlet any attempt <end>


---------- actual ----------
  smile please dont hesitate to get in touch with us off on pizzas on
online ordering only ovenstory pizza coupon code bmsosfifty validity m
ay may t c important instructions the coupons listed above are complet
ely owned by the brands who are providing the offer bookmyshow will no
t be liable if the brands refuse to accept the coupons at any brand ou
tlets please read all terms conditions of the respective coupon the co
upons listed above can be used only once at a brand outlet any attempt


---------- predicted ----------
```

Out[46]:

```
'dont hesitate to get <end> '
```

In [47]:

```
input_ = generate_sent(20)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  pizzas on online ordering only ovenstory pizza coupon code bmsosfifty vali
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont> hell
o yernagulahemanth bookmyshow dear customer your coupons have been lis
ted below if there is anything else we can do to make you smile please
dont hesitate to get in touch with us off on pizzas on online ordering
only ovenstory pizza coupon code bmsosfifty validity may may t c impor
tant instructions the coupons listed above are completely owned by the
brands who are providing the offer bookmyshow will not be liable if th
e brands refuse to accept the coupons at any brand outlets please read
all terms conditions of the respective coupon the coupons listed above
can be used only once at a brand outlet any attempt <end>


---------- actual ----------
  pizzas on online ordering only ovenstory pizza coupon code bmsosfift
y validity may may t c important instructions the coupons listed above
are completely owned by the brands who are providing the offer bookmys
how will not be liable if the brands refuse to accept the coupons at a
ny brand outlets please read all terms conditions of the respective co
upon the coupons listed above can be used only once at a brand outlet
any attempt


---------- predicted ----------
```

Out[47]:

```
'pizza coupon code bmsosfifty validity <end> '
```

In [48]:

```
input_ = generate_sent(20)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('   the brands who are providing the offer bookmyshow will not be liable if t
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> your coupons <cont> hell
o yernagulahemanth bookmyshow dear customer your coupons have been lis
ted below if there is anything else we can do to make you smile please
dont hesitate to get in touch with us off on pizzas on online ordering
only ovenstory pizza coupon code bmsosfifty validity may may t c impor
tant instructions the coupons listed above are completely owned by the
brands who are providing the offer bookmyshow will not be liable if th
e brands refuse to accept the coupons at any brand outlets please read
all terms conditions of the respective coupon the coupons listed above
can be used only once at a brand outlet any attempt <end>


---------- actual ----------
   the brands who are providing the offer bookmyshow will not be liabl
e if the brands refuse to accept the coupons at any brand outlets plea
se read all terms conditions of the respective coupon the coupons list
ed above can be used only once at a brand outlet any attempt


---------- predicted ----------
```

Out[48]:

```
'providing the offer bookmyshow <end> '
```

In [54]:

```
input_ = generate_sent(38)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' talked on the phone you asked me to use gcp utils gories you remember right
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> check this out <sub> regardi
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> check this out <sub> regarding dee
p learning project <cont> sir as we talked on the phone you asked me t
o use gcp utils gories you remember right even after using gcp the ram
<end>


---------- actual ----------
 talked on the phone you asked me to use gcp utils gories you remember
right even after using gcp the ram


---------- predicted ----------
```

Out[54]:

```
'talked on <end> '
```

In [55]:

```
input_ = generate_sent(38)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('  remember right even after using gcp the ram ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> check this out <sub> regardi
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> check this out <sub> regarding dee
p learning project <cont> sir as we talked on the phone you asked me t
o use gcp utils gories you remember right even after using gcp the ram
<end>


---------- actual ----------
  remember right even after using gcp the ram


---------- predicted ----------
```

Out[55]:

```
'gories you remember right <end> '
```

In [56]:

```
input_ = generate_sent(38)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' using gcp the ram ')
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> check this out <sub> regardi
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> check this out <sub> regarding dee
p learning project <cont> sir as we talked on the phone you asked me t
o use gcp utils gories you remember right even after using gcp the ram
<end>


---------- actual ----------
 using gcp the ram


---------- predicted ----------
```

Out[56]:

```
'using gcp <end> '
```

In [59]:

```
input_ = generate_sent(564)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print('looking forward to seeing you there during tomorrow s training we re going t
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> for our meeting to
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> for our meeting tomorrow
<cont> hello yernagulahemanth hey again ok so we ve finished our prepa
ration for tomorrow s workshop and we re really looking forward to see
ing you there during tomorrow s training we re going to be covering a
lot of topics ranging from important <end>


---------- actual ----------
looking forward to seeing you there during tomorrow s training we re g
oing to be covering a lot of topics ranging from important


---------- predicted ----------
```

Out[59]:

```
'to seeing you there during <end> '
```

In [61]:

```
input_ = generate_sent(564)
print('-'*10,'input','-'*10)
print(input_)
print('\n')
print('-'*10,'actual','-'*10)
print(' again ok so we ve finished our preparation for tomorrow s workshop and we r
print('\n')
print('-'*10,'predicted','-'*10)
predict_next_word('<start> <to> yernagulahemanth <prv> nan <sub> for our meeting to
```

```
---------- input ----------
<start> <to> yernagulahemanth <prv> nan <sub> for our meeting tomorrow
<cont> hello yernagulahemanth hey again ok so we ve finished our prepa
ration for tomorrow s workshop and we re really looking forward to see
ing you there during tomorrow s training we re going to be covering a
lot of topics ranging from important <end>


---------- actual ----------
 again ok so we ve finished our preparation for tomorrow s workshop an
d we re really looking forward to seeing you there during tomorrow s
training we re going to be covering a lot of topics ranging from impor
tant


---------- predicted ----------
```

Out[61]:

```
'again ok so <end> '
```

**Most of the predictions are correct, every time model id predicting 3 or more words**

NOTE: In actual section i have placed all next comming words(to make understad for readers) model only predicts minimum one word and maximum five words from the starting of actual section

# Steps Followed

**Getting Data**

1. Get data from personal emails(each email is stored in notepad) ref:Youtube (https://www.youtube.com/watch?v=l02tTpOPNro)
2. Extract to,subject,previous email(in case of replied email) and content and store them into a dataframe
3. Clean the data

**Preparing Data**

4. As shown in below table data is prepared

| Sentance | Output |
|---:|---:|
| This | is |
| This | is introduction |
| This | is introduction to |
| This | is introduction to my |
| This | is introduction to my project |
| This is | introduction |

| | |
|---|---|
| This is | introduction to |
| This is | introduction to my |
| This is | introduction to my |
| This is | introduction to my project |
| This is introduction | to |
| This is introduction | to my |
| This is introduction | to my project |

5. Now to the sentance part, I joined to, subject , previous email with there respective tags
   for example: < start > < to > yernagulahemanth < prv > nan < sub > for our meeting tomorrow < cont > hello yernagulahemanth hey again ok so we ve finished our preparation for tomorrow s workshop and we re really looking forward to seeing you there during tomorrow s training we re going to be covering a lot of topics ranging from important < end >
6. This sentance is fed into encoder to get hidden states and outputs
7. These hidden states and outputs are fed into attention layer to get context vector
8. This context vector passed to the decoder where final output is obtained

**For detailed explanation on model read this blog medium (https://medium.com/@yernagulahemanth/google-smart-compose-46b289eca6bc)**

# Reference:

1. https://arxiv.org/pdf/1906.00080.pdf (https://arxiv.org/pdf/1906.00080.pdf)
2. AppliedAi Course(Encoder-Decoder ) (https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4149/live-encoder-decoder-models/8/module-8-neural-networks-computer-vision-and-deep-learning)
3. AppliedAi Course(Attention Based Models) (https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4150/attention-models-in-deep-learning/8/module-8-neural-networks-computer-vision-and-deep-learning)
4. Floydhub(Detailed explination of attention based models (https://blog.floydhub.com/attention-mechanism/)
5. Language Translator with attention based model (https://towardsdatascience.com/intuitive-understanding-of-attention-mechanism-in-deep-learning-6c9482aecf4f)
6. Tensorflow implementation of attention based model (https://www.tensorflow.org/tutorials/text/nmt_with_attention)(Code Reference)
7. Machine Learning Mastery (https://machinelearningmastery.com/how-does-attention-work-in-encoder-decoder-recurrent-neural-networks/)
8. Coursera (https://www.coursera.org/lecture/nlp-sequence-models/attention-model-lSwVa)
9. Analytics Vidhya (https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/)

In [0]:

```
# sent_ = '<start> <to> team <prv> nan <sub> about mentor <cont> hello team i recen
# tokens = sent_.split()
# inputs_ = []
# to_pred = []
# but_pred= []
# new_q = ''
# for i,word in enumerate(tokens):
#     if i < len(tokens)-1:
#         inputs_.append(new_q)
#         to_pred.append(tokens[i+1])
#         new_q = new_q + ' ' + tokens[i+1]
#         pred = predict_next_word(u'<start> '+ new_q).replace('<end>','')

#         but_pred.append(pred)
# dd = pd.DataFrame()
# dd['inp'] = inputs_
# dd['to_pred'] = to_pred
# dd['but_pred']=but_pred

# # for i in range(dd.shape[0]):
# #     print(dd.inp.iloc[i]+' ::: '+dd.to_pred.iloc[i]+'  :::  '+dd.but_pred.iloc[


# dd
```