Machine Translation on Tellurian Lingo

Deep Learning Project

By

Eswar Chand Thokala – 002649408

Sharath Kumar Cherukuri – 002647840

Keshav Balivada – 002649998

Jyotsna Jakkam – 002662150

Sai Sirisha Pulaparthy – 002649050

Abstract:

Language is core medium of communication and translation is core tool for the understanding the information in unknown language. Machine translation helps the people to understand the information of unknown language without the help of Human translator. This project is brief introduction to machine translation. Machine translation is an excellent choice in situations where having the materials translated by a human translator would be too large an investment.

A computer program can translate tremendous amounts of content quickly. Even the most trustworthy employees can make mistakes. That's why, as a rule, the more people who have access to a document, the higher the security risks. By reducing the number of humans who must access sensitive data, machine translation can improve data security. In ancient days, German people suffered a lot with this translation issues. All the medical prescriptions were written in English, where German people can't understand. They must depend entirely on human translators. As no human can work accurately, sometimes the medicine name would be translated to other name. This resulted in wrong medication. So, these types of complex problems can be solved by using machine translation. This is the main motto to choose this project. Here we will be translating English to Hindi and German language using attention models as merely half of the world's population speak English, Hindi, and German. We also translate human readable dates like April 8th, 2000, to machine readable dates 04/08/2000 in this project.

CHAPTER 1
INTRODUCTION

Communication is important between humans to understand better by removing misunderstanding and creating clarity of thoughts and expression. There are many languages across the globe and each language has its own context. According to the research, 72.1% of the consumers spend most or all their time on sites in their own language. 72.4% say they would be more likely to buy a product with information in their own language. 56.2% say that the ability to obtain information in their own language is more important than price. Many people are finding difficulty in understanding text or any sought of information which is in other languages. People are finding it tougher to even understand medical prescriptions written in other languages, this is causing wrong diagnosis of people. Many advanced research papers were there in other native languages rather than being in English, this is creating a lot of technical gaps. At the same time, context is very important, if we lose context the whole meaning of the sentence changes is one more challenge we are conquering. Here, we are implementing real time communications where it would not be practical for a human to translate so accurately without losing the context of the sentence, because if context loses then the whole meaning changes.

While translating we consider emotions because emotions play a very important role in understanding. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication. We have built a neural network model which is well capable and unique to find these emotions. Context of the text. It will be able to translate the text from one language to another language with the help of emotions without losing the context. The implemented model is also well capable of identifying the emotion from an audio. The model is well able to deal with translation of text from one language to another and at the same time it also identifies and detects the emotions and context. Based on the emotion and context, the model will move to another language.

CHAPTER 2

NEURAL MACHINE TRANSLATION

The neural machine translation system (NMT) consists of an encoder-decoder architecture with one or more layers of RNN cells such as LSTM or GRU units. The sentences are encoded into the embedding of fixed dimension. Each word in the sentence is given as an input to the each RNN cell. Recurrent Neural Network (RNN) is used for various NLP tasks as they encode sequential information such as time series, sentences, speech etc. While training, there can be a problem of vanishing or exploding gradients while back propagating the error. But these problems can be dealt separately.

2.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) cells contains of three gates such as forget gate, input gate and output gate. Apart from the above three mentioned gates, LSTM cell takes an input from previous cell, retains a hid- den state. The diagram of LSTM cells is given in figure 1.

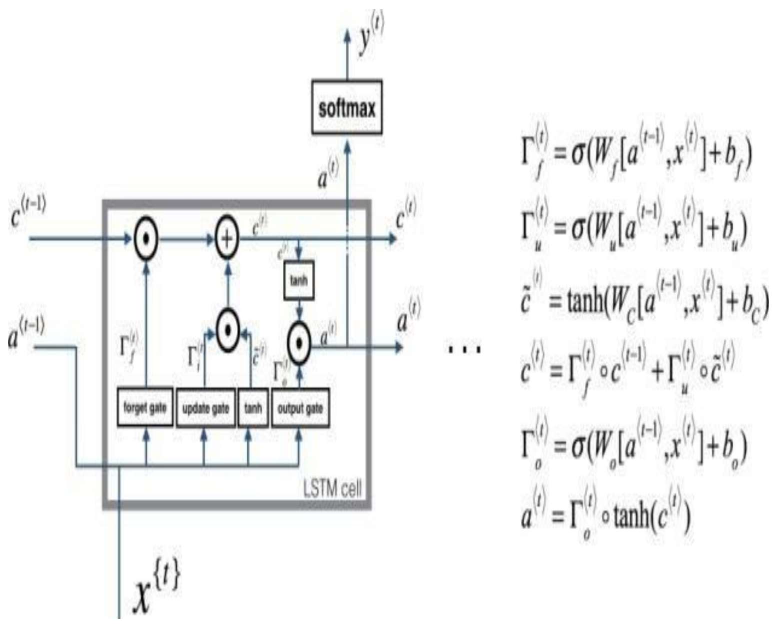
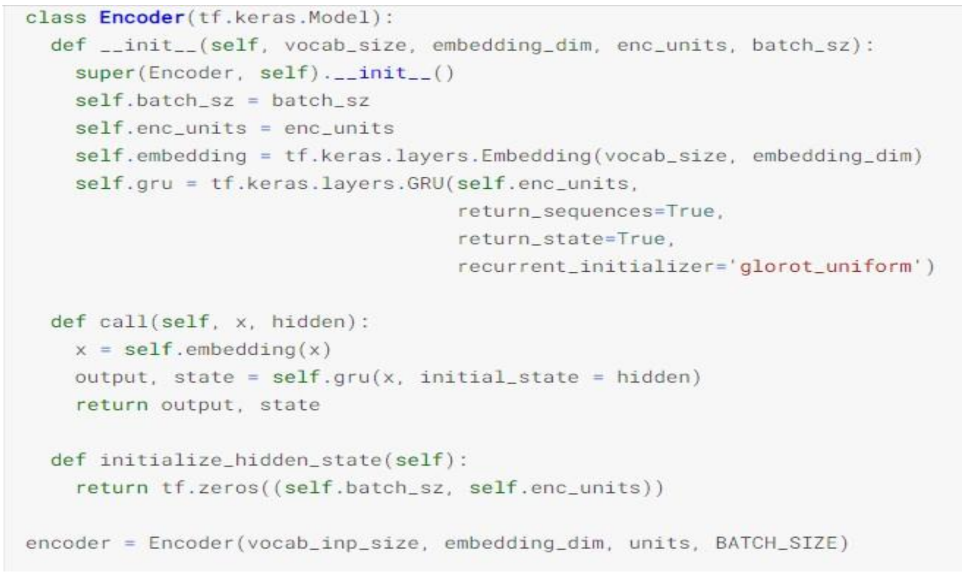

$y^{\langle t \rangle}$
softmax
$a^{\langle t \rangle}$
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
$a^{\langle t-1 \rangle}$
$\Gamma_f^{\langle t \rangle}$
$\Gamma_i^{\langle t \rangle}$
$\tilde{c}^{\langle t \rangle}$
$\Gamma_o^{\langle t \rangle}$
tanh
forget gate
update gate
tanh
output gate
LSTM cell
$x^{\{t\}}$
...
$\Gamma_f^{\langle t \rangle} = \sigma(W_f[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$
$\Gamma_u^{\langle t \rangle} = \sigma(W_u[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_u)$
$\tilde{c}^{\langle t \rangle} = \tanh(W_C[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_C)$
$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_u^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle}$
$\Gamma_o^{\langle t \rangle} = \sigma(W_o[a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$
$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh(c^{\langle t \rangle})$

Figure 1: A schematic diagram of LSTM cells taken from http://wiki.hacksmeta.com/ machine learning/deep learning/rnn-basics.html

2.2 Encoder Architecture

In NMT model, an encoder consists of a series of RNN cells in which each cell takes an input from each word embedding in the sentence as well has previous cell's hidden state. For the first cell, the hidden state is initialized with Xavier uniform initialization available in pytorch. In this way, each RNN cell's output is dependent on the current input word embedding as well as the hidden state of previous cell (In case of unidirectional). For bidirectional RNN, the output of each cell is also dependent on the future words in thesentence.

The implementation point of view, each sentence won't be of the same length. Hence the corpus is filtered out such that left over sentences have lengths less than or equal to some maximum length which is an hyperparameter. For smaller sentences, the sentences are padded with symbol'/PAD'. A batch of sentences are passed into the encoder and then decoded with decoder. Total number of layers are kept 4 as indicated in the paper. In this paper, the authors have suggested to keep the number of layers in between 2 to 4 for better results and speed. Here is the code snippet of Encoder we have used in the project.



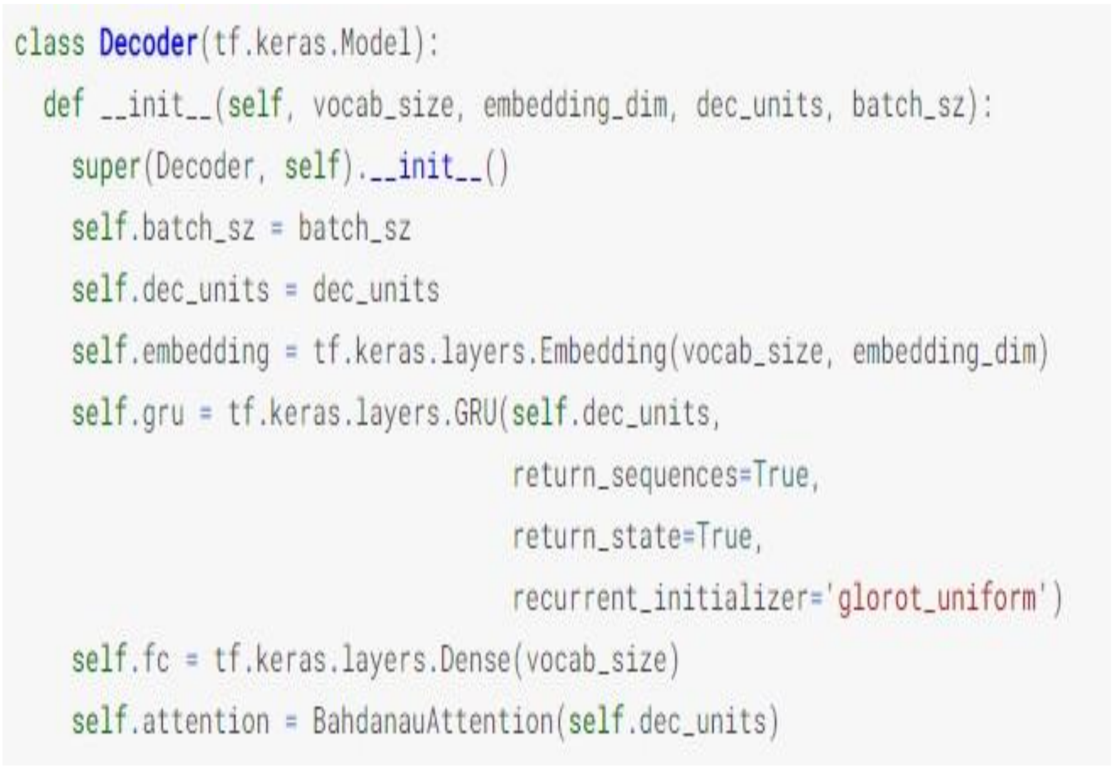
```
class Encoder(tf.keras.Model):
  def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
    super(Encoder, self).__init__()
    self.batch_sz = batch_sz
    self.enc_units = enc_units
    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(self.enc_units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')

  def call(self, x, hidden):
    x = self.embedding(x)
    output, state = self.gru(x, initial_state = hidden)
    return output, state

  def initialize_hidden_state(self):
    return tf.zeros((self.batch_sz, self.enc_units))

encoder = Encoder(vocab_inp_size, embedding_dim, units, BATCH_SIZE)
```

Figure 2: Encoder function code

From figure 2, we can see the code snippet of the Encoder Function written in the project. Here we have declared a function names Encoder which is used to encode the inputs. We have used Keras library in achieving this. We have used glorot_uniform as recurrent initialize.

2.3 Decoder Architecture

Similar to an encoder architecture. a decoder architecture also consists of some series of RNN cells with specified number of layers. While de- coding, an attention mechanism is used to get the words on which the decoder needs to give more weight to predict the current the word given the previous words. For each time steps while de- coding, the attention weights are calculated with methods such as additive attention (Bahdanau et al.,2015), multiplicative attention (Luong et al., 2015), Key-value attention (Liu and Lapata,2017) etc. These mechanisms have been discussed in the next section. Here is the Decoder function code snippet which we have used in the project.



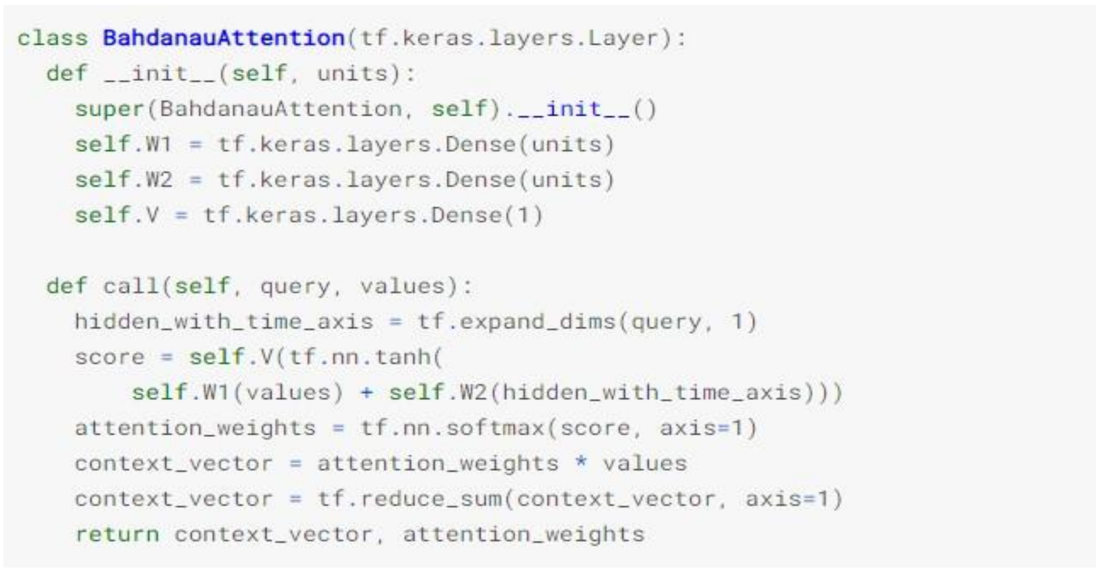
```
class Decoder(tf.keras.Model):
  def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
    super(Decoder, self).__init__()
    self.batch_sz = batch_sz
    self.dec_units = dec_units
    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(self.dec_units,
                                   return_sequences=True,
                                   return_state=True,
                                   recurrent_initializer='glorot_uniform')
    self.fc = tf.keras.layers.Dense(vocab_size)
    self.attention = BahdanauAttention(self.dec_units)
```

Figure 3 : Decoder Function

From figure 3, we can see the code snippet of the Decoder Function written in the project. Here we have declared a function names Decoder which is used to decode the encoded version. We have used Keras library in achieving this. We have used glorot_uniform as recurrent initialize.

2.4 Attention Mechanism

Basic attention mechanisms as well as its motive have been discussed earlier. But in various attention schemes, the formulae in which the attention weights are calculated are different. For brevity, various attention weights formulae are given as follows. Here s_j and h_i are encoder state and cur- rent hidden state respectively.



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

  def call(self, query, values):
    hidden_with_time_axis = tf.expand_dims(query, 1)
    score = self.V(tf.nn.tanh(
        self.W1(values) + self.W2(hidden_with_time_axis)))
    attention_weights = tf.nn.softmax(score, axis=1)
    context_vector = attention_weights * values
    context_vector = tf.reduce_sum(context_vector, axis=1)
    return context_vector, attention_weights
```

Figure 4: Attention Function

From Figure 4 we can see the code snippet of the Attention Mechanism we have implemented in the project. We have created 3 dense layers using tensorflow. We have also initialized weights for the neurons in the hidden layers using keras. In this way we have built the attention mechanism.

CHAPTER 3

Implementation Details

Context is very important, if we lose context the whole meaning of the sentence changes is one more challenge we are conquering. Here, we are implementing real time communications where it would not be practical for a human to translate so accurately without losing context. While translating we consider emotions because emotions play a very important role in understanding. As emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today's digital world of remote communication. We have built a neural network model which is well capable and unique to find these emotions in the context of the text. It will be able to translate the text from one language to another language with the help of emotions without losing the context. The implemented model is also well capable of identifying the emotion from an audio. The model is well able to deal with translation of text from one language to another and at the same time it also identifies and detects the emotions and context. Based on the emotion and context, the model will move to another language.

RNNs are also capable of doing natural language translation, aka. machine translation. It involves two RNNs, one for the source language and one for the target language. One of them is called an encoder, and the other one decoder. The reason is that the first one encodes the sentence into a vector and the second one converts the encoded vector into a sentence in target language. The decoder is a separate RNN. Given the encoded sentence, it produces the translated sentence in target language. Attention lets the decoder to focus on specific parts of the input sentence for each output word. This helps the input and output sentences to align with one another.

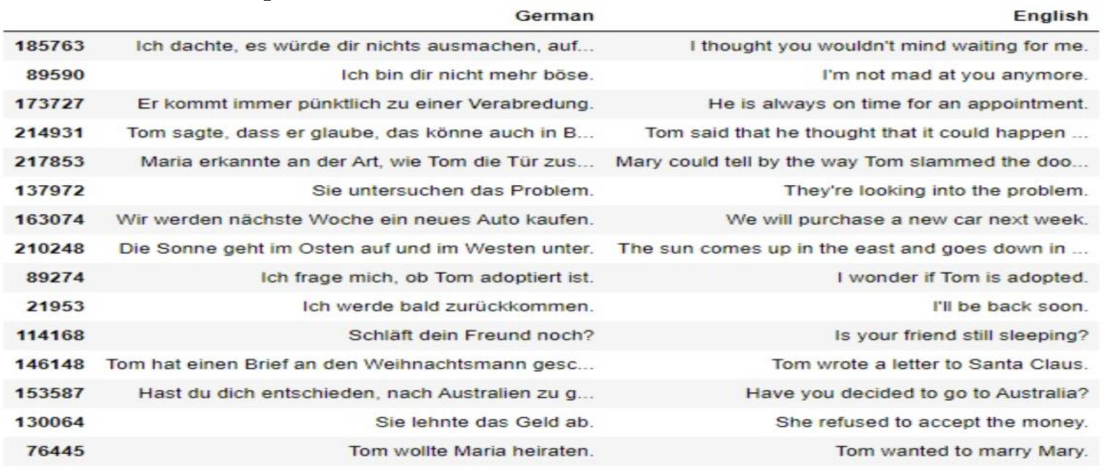
CHAPTER 4

German to English Translation

4.1 Dataset

Due to lack of enough computational power, an NMT model is trained on the dataset given at a site http://www.manythings.org/anki/deu-eng.zip. Here two separate files are created for both English as well as German sentences. Out of these, maximum length of sentences is limited to 25 and minimum 3. Total sentences taken for training are 1,00,000. There are three columns in the dataset. First column contains RowID, which is unique and given to each and every sentence. Second column consists of English sentences. Third column consists of German sentences. Now we will train our model such that, our model is well capable in translating any sentence from English to German without losing its context.

Here is the sample of the dataset we have used.



	German	English
185763	Ich dachte, es würde dir nichts ausmachen, auf...	I thought you wouldn't mind waiting for me.
89590	Ich bin dir nicht mehr böse.	I'm not mad at you anymore.
173727	Er kommt immer pünktlich zu einer Verabredung.	He is always on time for an appointment.
214931	Tom sagte, dass er glaube, das könne auch in B...	Tom said that he thought that it could happen ...
217853	Maria erkannte an der Art, wie Tom die Tür zus...	Mary could tell by the way Tom slammed the doo...
137972	Sie untersuchen das Problem.	They're looking into the problem.
163074	Wir werden nächste Woche ein neues Auto kaufen.	We will purchase a new car next week.
210248	Die Sonne geht im Osten auf und im Westen unter.	The sun comes up in the east and goes down in ...
89274	Ich frage mich, ob Tom adoptiert ist.	I wonder if Tom is adopted.
21953	Ich werde bald zurückkommen.	I'll be back soon.
114168	Schläft dein Freund noch?	Is your friend still sleeping?
146148	Tom hat einen Brief an den Weihnachtsmann gesc...	Tom wrote a letter to Santa Claus.
153587	Hast du dich entschieden, nach Australien zu g...	Have you decided to go to Australia?
130064	Sie lehnte das Geld ab.	She refused to accept the money.
76445	Tom wollte Maria heiraten.	Tom wanted to marry Mary.

Figure: 5 Dataset sample for German to English Translation

From the figure 5, we can see the sample of the dataset used in English to German translation. There are three columns in the dataset. First column contains RowID, which is unique and given to each and every sentence. Second column consists of German sentences. Third column consists of English sentences

4.2 Architecture Details

Here both encoder and decoder layers use 4 layers of bidirectional LSTM. The training is done batch- wise with a batch size of 256. Embedding size is kept as 128 as given in embedding dimension doesn't add much for BLEU score though 2048 achieved best results according to the paper. Hidden size dimension is kept as 512, learning rate as 0.001. The training is via ADAM optimizer with 20 epochs of 1 Lakhs sentences. The gradients have been clipped once they go above 50.

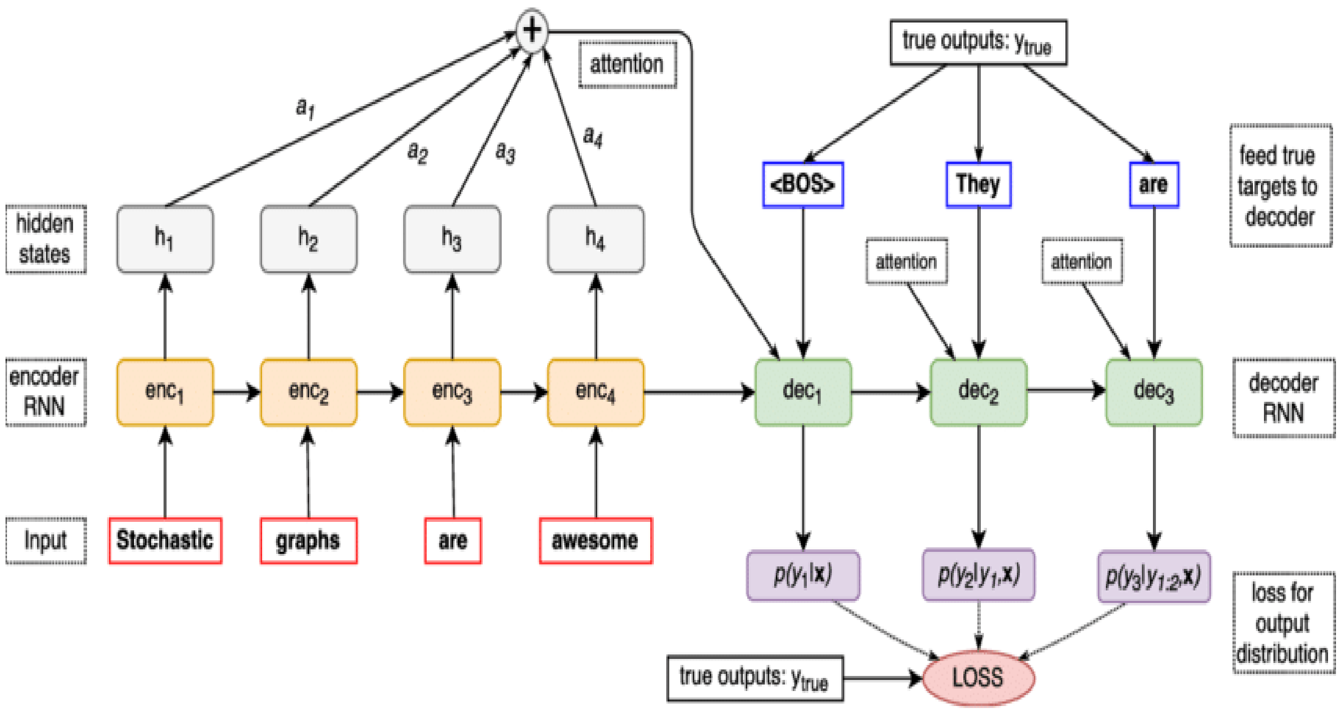


Fig : Encoder and Decoder with Attention Model Architecture

A sequence to sequence model has two parts – an encoder and a decoder. Both the parts are practically two different neural network models combined into one giant network. the task of an encoder network is to understand the input sequence, and create a smaller dimensional representation of it. This representation is then forwarded to a decoder network which generates a sequence of its own that represents the output. The input is put through an encoder model which gives us the encoder output. Here, each input words is assigned a weight by the attention mechanism which is then used by the decoder to predict the next word in the sentence. We use Bahdanau attention for the encoder.

4.3 Training

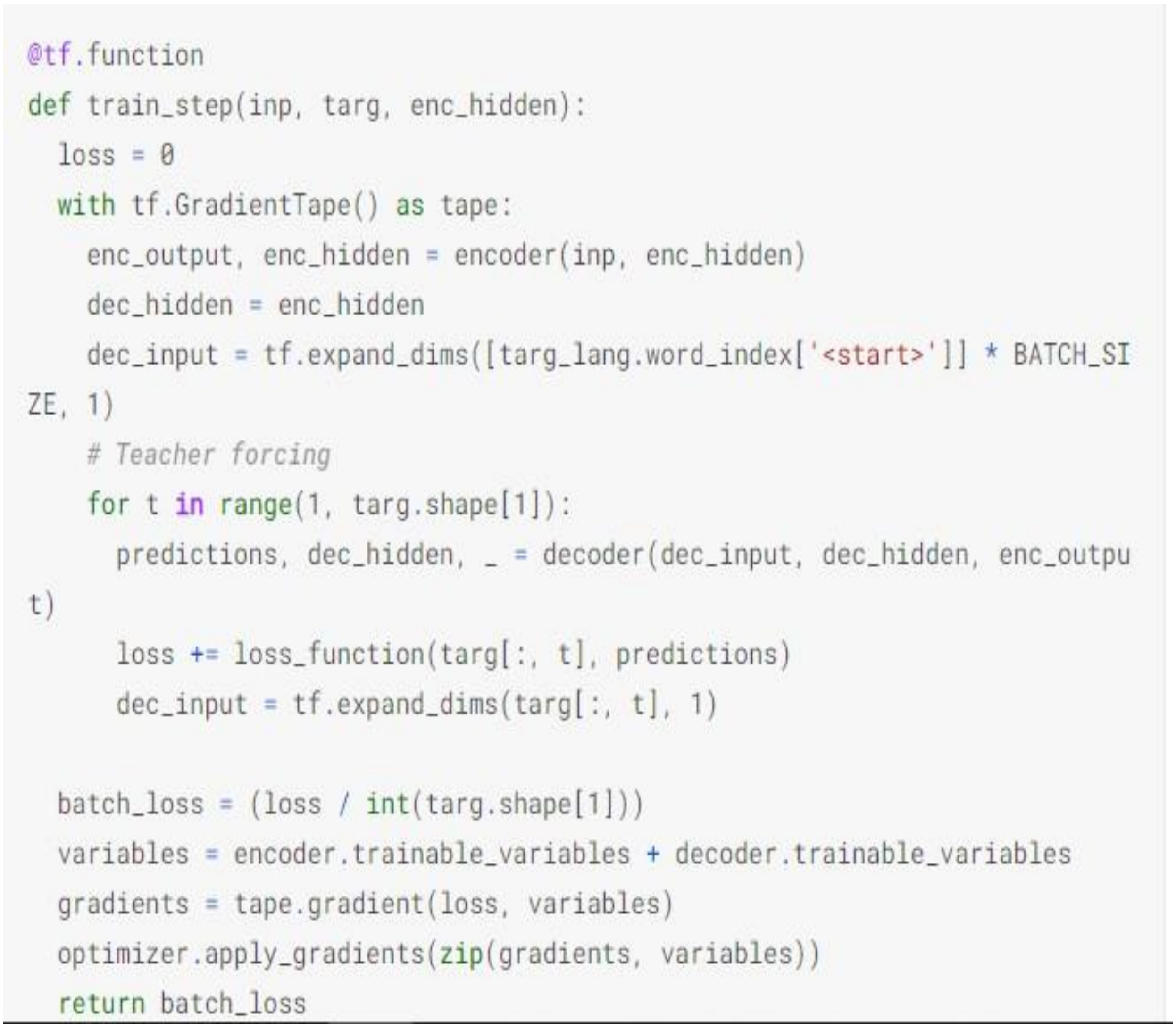
With the complete model in hand, we can now take a closer look at training. One challenge is that the number of steps in the decoder and the number of steps in the encoder varies with each training example. Sentence pairs consist of sentences of different length, so we cannot have the same computation graph for each training example but instead have to dynamically create the computation graph for each of them. This technique is called unrolling the recurrent neural networks, and we already discussed it with regard to language models

Practical training of neural machine translation models requires GPUs which are well suited to the high degree of parallelism inherent in these deep learning models (just think of the many matrix multiplications). To increase parallelism even more, we process several sentence pairs (say, 100) at once. This implies that we increase the dimensionality of all the state tensors. To given an example. We represent each input word in specific sentence pair with a vector hj . Since we already have a sequence of input words, these are lined up in a matrix. When we process a batch of sentence pairs, we again line up these matrices into a 3-dimensional tensor. Similarly, to give another example, the decoder hidden state si is a vector for each output word. Since we process a batch of sentences, we line up their hidden states into a matrix. Note that in this case it is not helpful to line up the states for all the output words, since the states are computed sequentially.

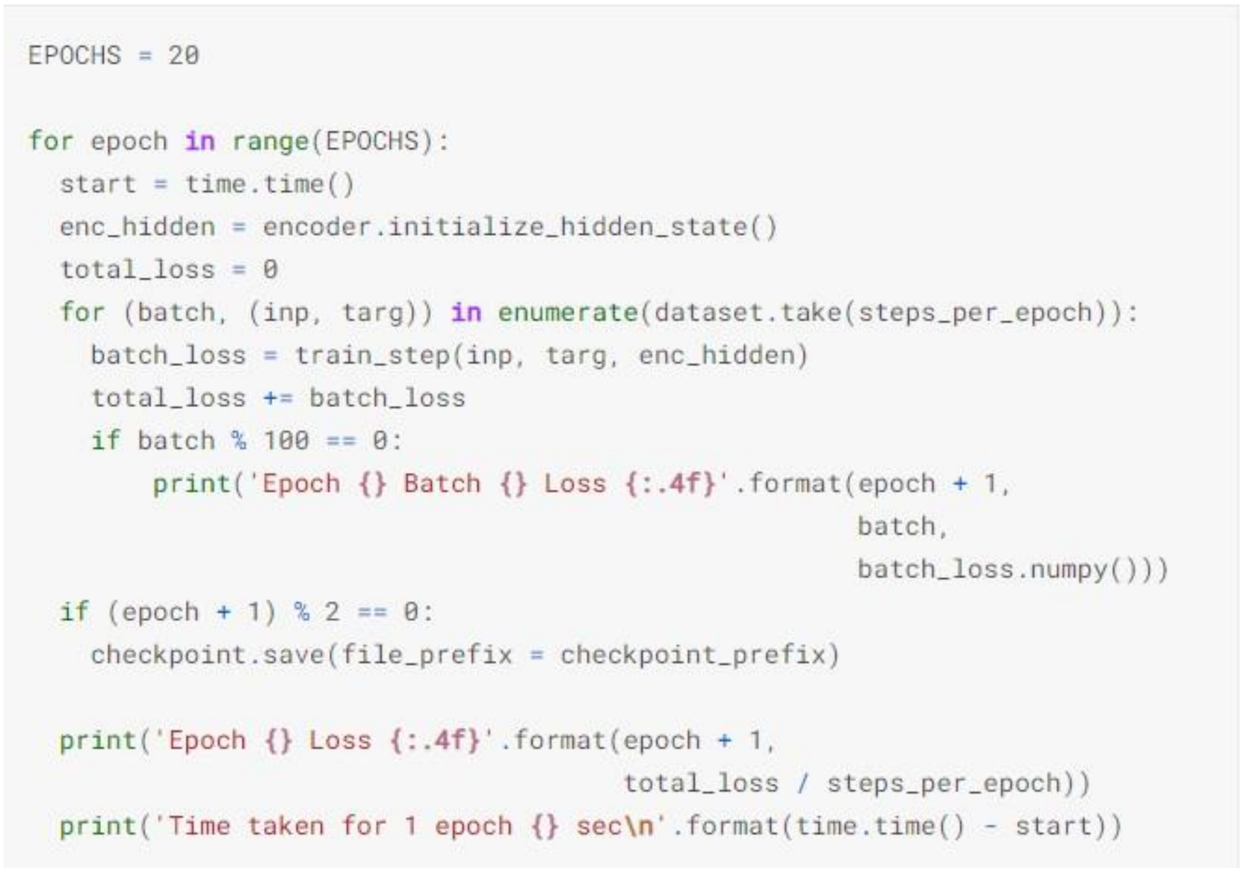
Here are the steps involved in the training of the model:

1. Pass input through encoder to get encoder output.
2. Then encoder output, encoder hidden state and the decoder input is passed to decoder.
3. Decoder returns predictions and decoder hidden state.
4. Decoder hidden state is then passed back to model.
5. Predictions are used to calculate loss.
6. Use teacher forcing (technique where the target word is passed as the next input to the decoder) for the next input to the decoder.
7. Calculate gradients and apply it to optimizer for back propogation.

The below are the code snippets used in the project to train the model in order to translate from one language to another that is from English language to German language.



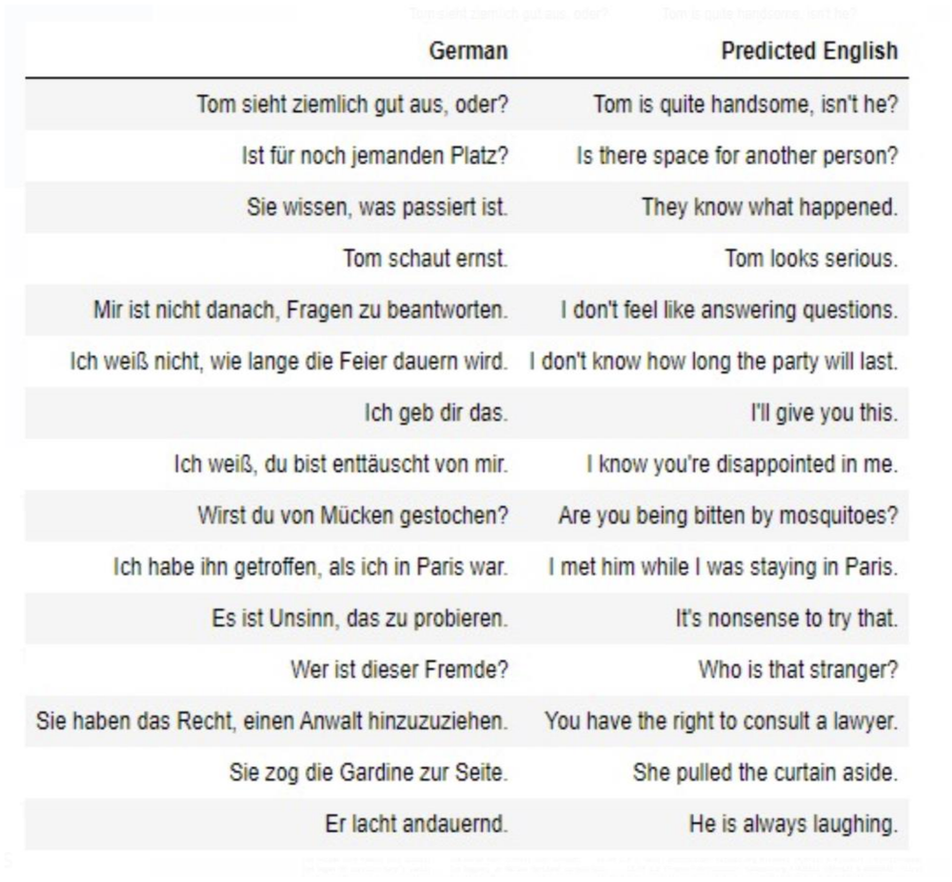
```
@tf.function
def train_step(inp, targ, enc_hidden):
  loss = 0
  with tf.GradientTape() as tape:
    enc_output, enc_hidden = encoder(inp, enc_hidden)
    dec_hidden = enc_hidden
    dec_input = tf.expand_dims([targ_lang.word_index['<start>']] * BATCH_SI
ZE, 1)
    # Teacher forcing
    for t in range(1, targ.shape[1]):
      predictions, dec_hidden, _ = decoder(dec_input, dec_hidden, enc_outpu
t)
      loss += loss_function(targ[:, t], predictions)
      dec_input = tf.expand_dims(targ[:, t], 1)

  batch_loss = (loss / int(targ.shape[1]))
  variables = encoder.trainable_variables + decoder.trainable_variables
  gradients = tape.gradient(loss, variables)
  optimizer.apply_gradients(zip(gradients, variables))
  return batch_loss
```


```
EPOCHS = 20

for epoch in range(EPOCHS):
  start = time.time()
  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
    if batch % 100 == 0:
        print('Epoch {} Batch {} Loss {:.4f}'.format(epoch + 1,
                                                     batch,
                                                     batch_loss.numpy()))
  if (epoch + 1) % 2 == 0:
    checkpoint.save(file_prefix = checkpoint_prefix)

  print('Epoch {} Loss {:.4f}'.format(epoch + 1,
                                      total_loss / steps_per_epoch))
  print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```
Figure 12: Code snippet of model training.

From Figure 12, it is understandable that the code is written to train the model. Here we have taken 20 epochs. For each epoch the model will see the complete dataset once. More the epochs the better is the model. We have taken 400 batches. For each epoch the model calculates the loss for every 100 batches. For every epoch, the loss is calculated for every 100 batches i.e at 0, 100, 200, 300 and 400. At last the model also updates total time taken for 1 epoch. The same procedure continues until it completes all the epochs.

4.4 Predictions

Here are some of the predictions done by the model.



German	Predicted English
Tom sieht ziemlich gut aus, oder?	Tom is quite handsome, isn't he?
Ist für noch jemanden Platz?	Is there space for another person?
Sie wissen, was passiert ist.	They know what happened.
Tom schaut ernst.	Tom looks serious.
Mir ist nicht danach, Fragen zu beantworten.	I don't feel like answering questions.
Ich weiß nicht, wie lange die Feier dauern wird.	I don't know how long the party will last.
Ich geb dir das.	I'll give you this.
Ich weiß, du bist enttäuscht von mir.	I know you're disappointed in me.
Wirst du von Mücken gestochen?	Are you being bitten by mosquitoes?
Ich habe ihn getroffen, als ich in Paris war.	I met him while I was staying in Paris.
Es ist Unsinn, das zu probieren.	It's nonsense to try that.
Wer ist dieser Fremde?	Who is that stranger?
Sie haben das Recht, einen Anwalt hinzuzuziehen.	You have the right to consult a lawyer.
Sie zog die Gardine zur Seite.	She pulled the curtain aside.
Er lacht andauernd.	He is always laughing.

Figure 6: Predictions by the model

From Figure 6, we can see the translations of sentences from English language to German language. The sentence ‘You run’ in English is translated to the sentence ‘ Du Laufst’ which is German. Similarly the sentence ‘Be Brave’ in English is translated to the sentence ‘ Sei tapfer’ which is in German. In this way, above shown sentences are translated into German language accurately.

The predictions are so accurate and also cross checked with the google translator. Below are some of the predictions generated by the model.

- **Source :** bringen das dieses buch

  Prediction: take this book back to him

- **Source:** sie hat sich um ihrem

  Prediction: she looked after her baby

- **Source :** er ging irgendwohin

  Prediction: he went to some place or other Truth

- **Source :** ich habe tom schon alles gesagtnicht

  Prediction : i already told tom everything

As length of the sentence increases, the prediction gets deteriorated. While the shorter sentences are translated more accurately.

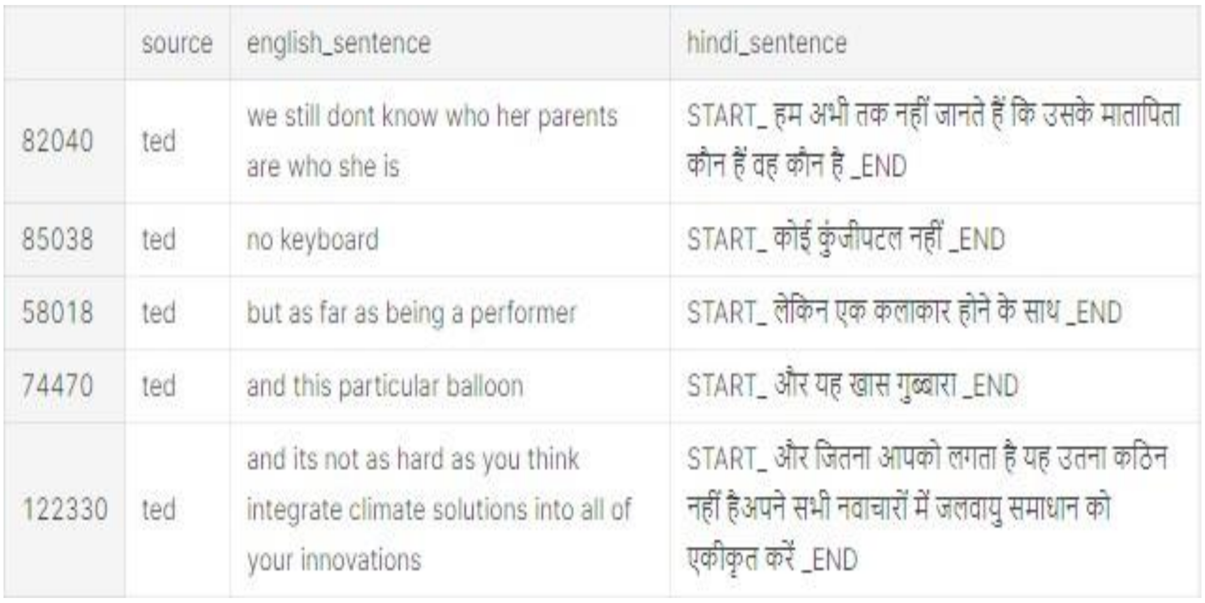
CHAPTER 5

English to Hindi Translation

5.1 Dataset

An NMT model is trained on the dataset given in the assignment. Here two separate files are created for both English as well as German sentences. Out of these, maximum length of sentences is limited to 25 and minimum 3. Total sentences taken for training are 10,000. Here due to tokenization issue, the GPU memory usage was exceeding GPU capacity.There are three columns in the dataset. First column contains source, which mentions from where certain sentences have been taken. Second column consists of English sentences. Third column consists of Hindi sentences. Now we will train our model such that, our model is well capable in translating any sentence from English to Hindi without losing its context.

Here is the Sample of the dataset.



	source	english_sentence	hindi_sentence
82040	ted	we still dont know who her parents are who she is	START_ हम अभी तक नहीं जानते हैं कि उसके मातापिता कौन हैं वह कौन है _END
85038	ted	no keyboard	START_ कोई कुंजीपटल नहीं _END
58018	ted	but as far as being a performer	START_ लेकिन एक कलाकार होने के साथ _END
74470	ted	and this particular balloon	START_ और यह खास गुब्बारा _END
122330	ted	and its not as hard as you think integrate climate solutions into all of your innovations	START_ और जितना आपको लगता है यह उतना कठिन नहीं हैअपने सभी नवाचारों में जलवायु समाधान को एकीकृत करें _END

Figure 11: English to Hindi Dataset.

From the figure 11, we can see the sample of the dataset used in English to German translation. There are three columns in the dataset. First column contains source, which mentions from where certain sentences have been taken. Second column consists of English sentences. Third column consists of Hindi sentences. Now we will train our model such that, our model is well capable in translating any sentence from English to Hindi without losing its context.

5.2 Architecture Details

Here both encoder and decoder layers use 4 layers of bidirectional LSTM. The training is done batch-wise with a batch size of 128. Embedding size is kept as 128 as given. Hidden size dimension is kept as 512, learning rate as 0.001. The training is via ADAM optimizer with 50 epochs of 10,000 sentences. The gradients have been clipped once they go above 50. In the assignment, the focus has been kept on being able to translate few sentences but as much as good quality as possible. Hence in both language translations, few sentences has been picked than mentioned and more epochs has been run to ensure we get good translation quality at least on training data.

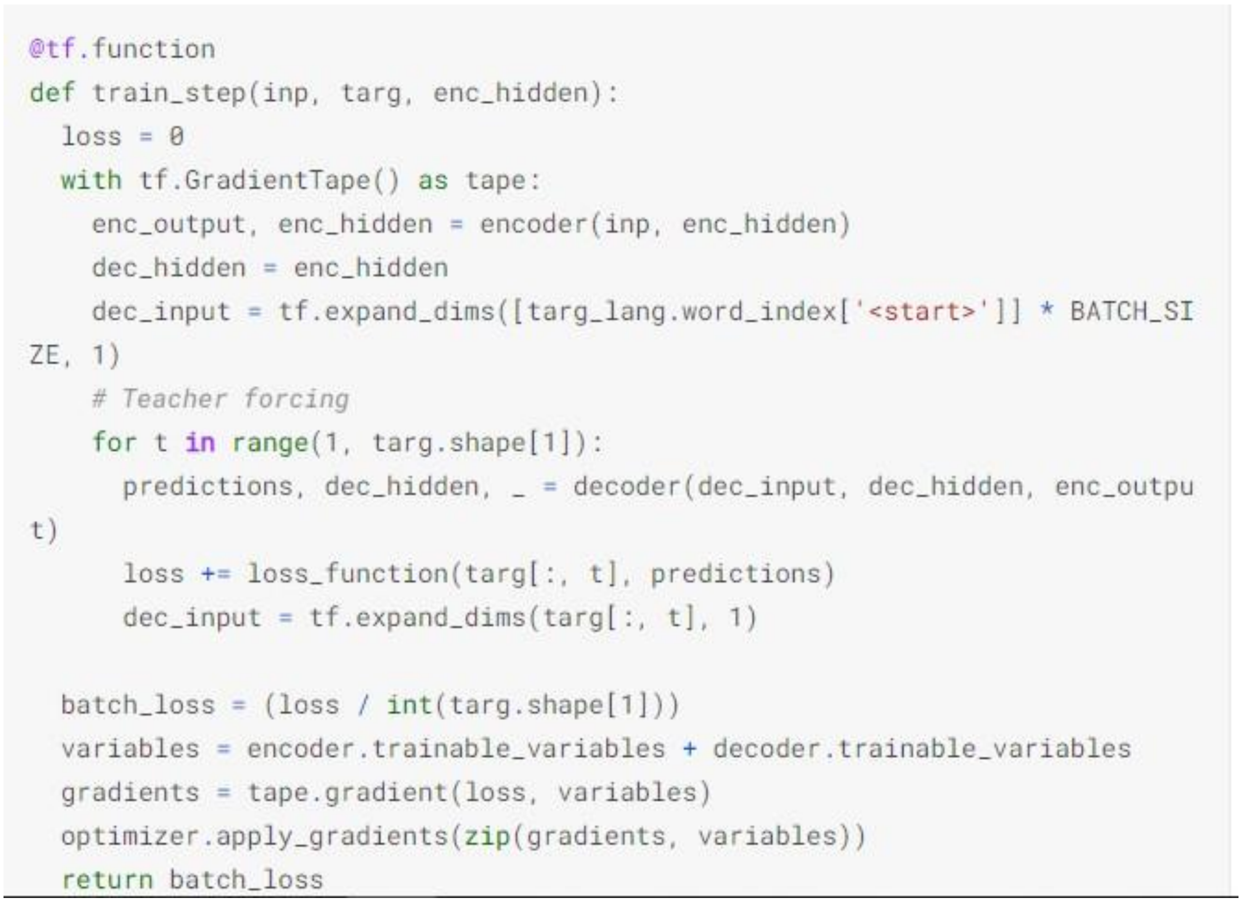
5.3 Training

With the complete model in hand, we can now take a closer look at training. One challenge is that the number of steps in the decoder and the number of steps in the encoder varies with each training example. Sentence pairs consist of sentences of different length, so we cannot have the same computation graph for each training example but instead have to dynamically create the computation graph for each of them. This technique is called unrolling the recurrent neural networks, and we already discussed it with regard to language models

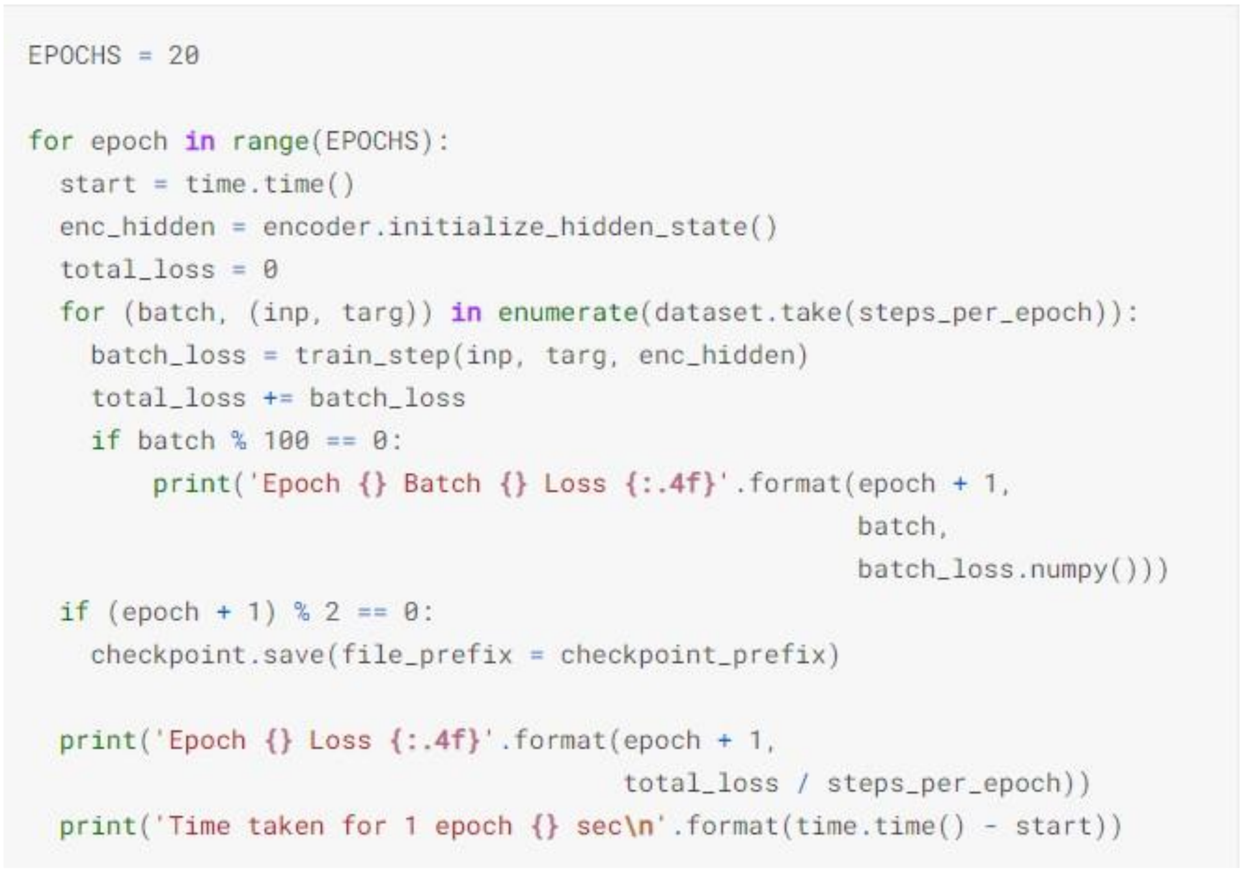
Practical training of neural machine translation models requires GPUs which are well suited to the high degree of parallelism inherent in these deep learning models (just think of the many matrix multiplications). To increase parallelism even more, we process several sentence pairs (say, 100) at once. This implies that we increase the dimensionality of all the state tensors. To given an example. We represent each input word in specific sentence pair with a vector hj . Since we already have a sequence of input words, these are lined up in a matrix. When we process a batch of sentence pairs, we again line up these matrices into a 3-dimensional tensor. Similarly, to give another example, the decoder hidden state si is a vector for each output word. Since we process a batch of sentences, we line up their hidden states into a matrix. Note that in this case it is not helpful to line up the states for all the output words, since the states are computed sequentially.

Here are the steps involved in the training of the model:

1. Pass input through encoder to get encoder output.
2. Then encoder output, encoder hidden state and the decoder input is passed to decoder.
3. Decoder returns predictions and decoder hidden state.
4. Decoder hidden state is then passed back to model.
5. Predictions are used to calculate loss.
6. Use teacher forcing (technique where the target word is passed as the next input to the decoder) for the next input to the decoder.
7. Calculate gradients and apply it to optimizer for backpropogation.



```
@tf.function
def train_step(inp, targ, enc_hidden):
  loss = 0
  with tf.GradientTape() as tape:
    enc_output, enc_hidden = encoder(inp, enc_hidden)
    dec_hidden = enc_hidden
    dec_input = tf.expand_dims([targ_lang.word_index['<start>']] * BATCH_SI
ZE, 1)
    # Teacher forcing
    for t in range(1, targ.shape[1]):
      predictions, dec_hidden, _ = decoder(dec_input, dec_hidden, enc_outpu
t)
      loss += loss_function(targ[:, t], predictions)
      dec_input = tf.expand_dims(targ[:, t], 1)

  batch_loss = (loss / int(targ.shape[1]))
  variables = encoder.trainable_variables + decoder.trainable_variables
  gradients = tape.gradient(loss, variables)
  optimizer.apply_gradients(zip(gradients, variables))
  return batch_loss
```



```
EPOCHS = 20

for epoch in range(EPOCHS):
  start = time.time()
  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
    if batch % 100 == 0:
        print('Epoch {} Batch {} Loss {:.4f}'.format(epoch + 1,
                                                     batch,
                                                     batch_loss.numpy()))

  if (epoch + 1) % 2 == 0:
    checkpoint.save(file_prefix = checkpoint_prefix)

  print('Epoch {} Loss {:.4f}'.format(epoch + 1,
                                      total_loss / steps_per_epoch))
  print('Time taken for 1 epoch {} sec\n'.format(time.time() - start))
```

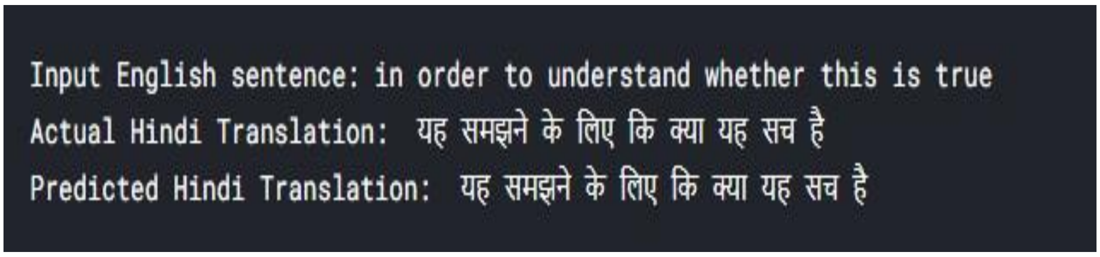
Figure 12: Code snippet of model training.

From Figure 12, it is understandable that the code is written to train the model. Here we have taken 20 epochs. For each epoch the model will see the complete dataset once. More the epochs the better is the model. We have taken 400 batches. For each epoch the model calculates the loss for every 100 batches. For every epoch, the loss is calculated for every 100 batches i.e at 0, 100, 200, 300 and 400. At last the model also updates total time taken for 1 epoch. The same procedure continues until it completes all the epochs.

5.4 Results

Here are some of the predictions of the model. Let us look into some examples just to evaluate the model performance, whether it is translating correctly or not.

Example 1:

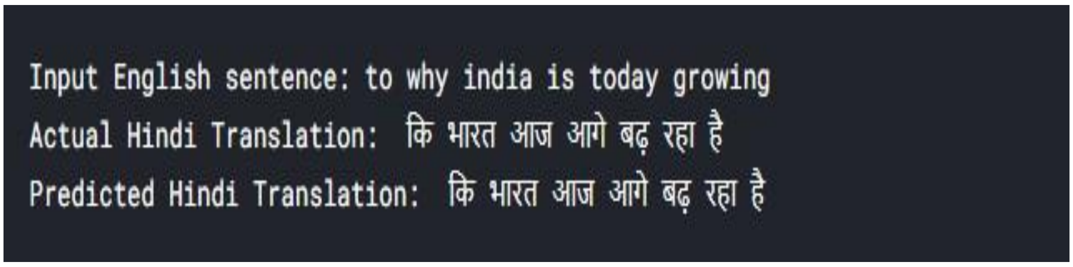


```
Input English sentence: in order to understand whether this is true
Actual Hindi Translation:  यह समझने के लिए कि क्या यह सच है
Predicted Hindi Translation:  यह समझने के लिए कि क्या यह सच है
```

Figure 12: Translated sentence from English to Hindi example 1

From figure 12, it is evident that the Input sentence ‘ In order to understand whether this is true’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

Example 2:

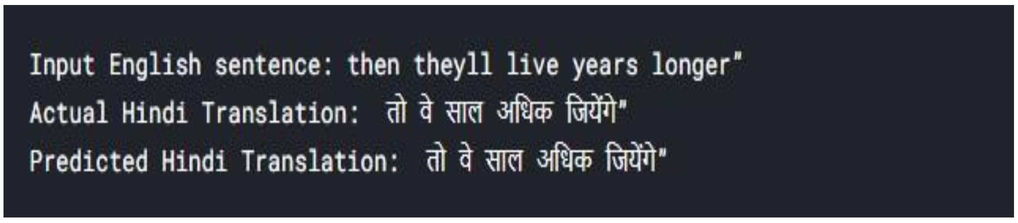


```
Input English sentence: to why india is today growing
Actual Hindi Translation:  कि भारत आज आगे बढ़ रहा है
Predicted Hindi Translation:  कि भारत आज आगे बढ़ रहा है
```

Figure 13: Translated sentence from English to Hindi example 2

From figure 13, it is evident that the Input sentence ‘To why India is today growing’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

Example 3:

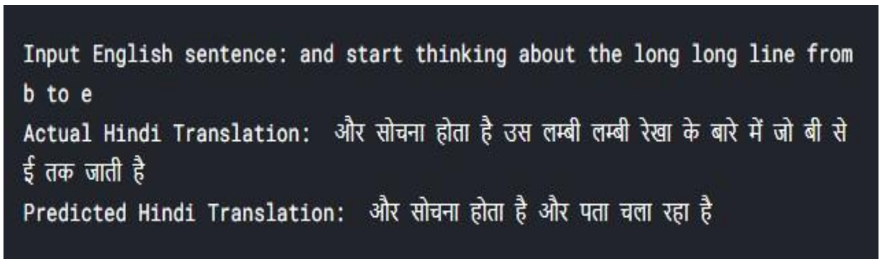


```
Input English sentence: then theyll live years longer"
Actual Hindi Translation:  तो वे साल अधिक जियेंगे"
Predicted Hindi Translation:  तो वे साल अधिक जियेंगे"
```

Figure 14: Translated sentence from English to Hindi example 3

From figure 14, it is evident that the Input sentence ‘They they’ll live years longer’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

Example 4:

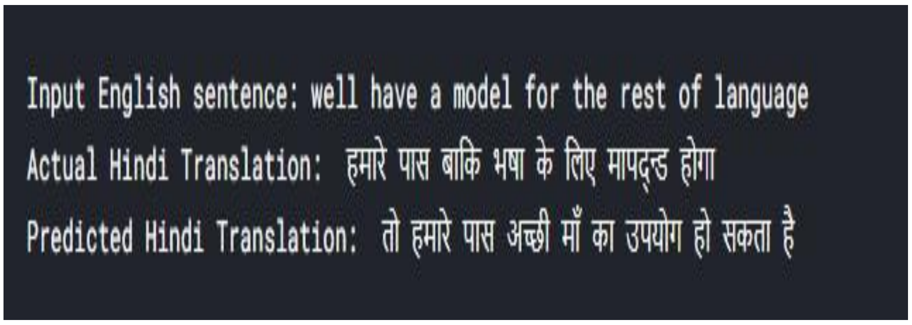


```
Input English sentence: and start thinking about the long long line from
b to e
Actual Hindi Translation:  और सोचना होता है उस लम्बी लम्बी रेखा के बारे में जो बी से
ई तक जाती है
Predicted Hindi Translation:  और सोचना होता है और पता चला रहा है
```

Figure 15: Translated sentence from English to Hindi example 4

From figure 15, it is evident that the Input sentence ‘And start thinking about the long line from b to e’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

Example 5:

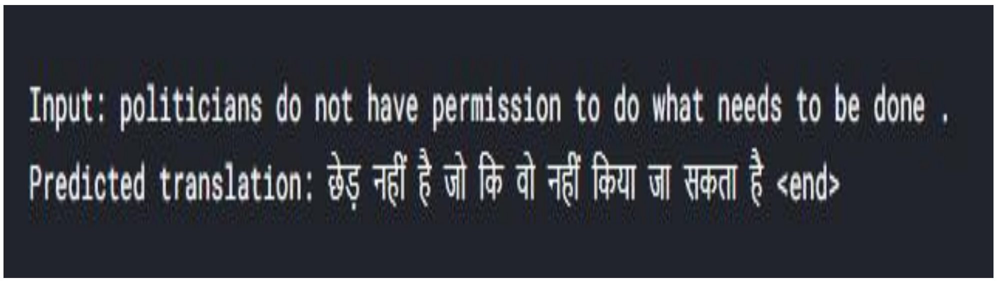


```
Input English sentence: well have a model for the rest of language
Actual Hindi Translation:  हमारे पास बाकि भषा के लिए मापद्न्ड होगा
Predicted Hindi Translation:  तो हमारे पास अच्छी माँ का उपयोग हो सकता है
```

Figure 16: Translated sentence from English to Hindi example 5

From figure 16, it is evident that the Input sentence ‘Well have a model for the rest of language ’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

Example 6:



```
Input: politicians do not have permission to do what needs to be done .
Predicted translation: छेड़ नहीं है जो कि वो नहीं किया जा सकता है <end>
```

Figure 18: Translated sentence from English to Hindi example 6

From figure 18, it is evident that the Input sentence ‘Do not have permission to do what needs to be done’ which is in English Language is accurately translated into Hindi Language. The sentences of actual Hindi sentence and predicted Hindi sentence are both same. By this we can understand that model is so accurate in translating from one language to another i.e English language to Hindi language. As the model does not lose the context, the output obtained by the model is so accurate.

7. References

[1]. http://www.unitedlanguagegroup.com/

[2]. http://www.readbag.com/

[3]. https://machinelearningmastery.com/develop-neural-machine-translation-system-keras/

[4]. https://www.dataversity.net/neural-machine-translation-with-sequence-to-sequence-rnn/

[5]. https://stackabuse.com/python-for-nlp-neural-machine-translation-with-seq2seq-in-keras/

[6]. https://paperswithcode.com/task/machine-translation