English to हिन्दी (Hindi) machine translation with multiple models.

By : Ram Nikhilesh Mekala , Zeeshan Ahmed Lodhia

Guided by: Mohammad Aminul Islam

Index

Introduction

Machine Translation in layman terms a translator which is a machine, it's about translating one language which could be the language which we know and the machine will translate it to something which we need or require. Machine translation is a wide scope but here in this project we are focusing on English to hindi translation, we are trying using different architecture and understand which one works better in all the different architecture. We explore various transformer models with and without attention which could be used for machine translation between English to Hindi sentences. Architectures such as sequence to sequence machine translation with the LSTM network, the other architecture is like the transformer which has encoder and decoder architecture further we try to use the model with encoder to decoder with attention. Before jumping to high technical stuff we need to know what is machine translation and how we can use it for our project. We conduct a substantial hyper parameter search to optimize our model's performance, on our dataset. We modify and test parameters including attention type (dot product, additive, or none), dropout rate, depth (number of layers), and using a bidirectional encoder. After all the analysis we came to a understand the transformer with attention proposed by bahdanau has a bit better performance than the one with no attention.

1.1 Machine Translation

Machine translation (MT) is automated translation. It is the process by which computer software is used to translate a text from one natural language (such as English) to another (such as Hindi).

To process any translation, human or automated, the meaning of a text in the original (source) language must be fully restored in the target language, i.e. the translation. While on the surface this seems straightforward, it is far more complex. Translation is not a mere word-for-word substitution. A translator must interpret and analyze all of the elements in the text and know how each word may influence another. This requires extensive expertise in grammar, syntax (sentence structure), semantics (meanings), etc., in the source and target languages, as well as familiarity with each local region.

Human and machine translation each have their share of challenges. For example, no two individual translators can produce identical translations of the same text in the same language pair, and it may take several rounds of revisions to meet customer satisfaction. But the greater challenge lies in how machine translation can produce publishable quality translations.

1.2 Encoder

A stack of several recurrent units (LSTM or GRU cells for better performance) where each accepts a single element of the input sequence, collects information for that element and propagates it forward. That is what an encoder does, it's function is mainly to process the input, generate encodings, which would be having information about the parts of the inputs and how it is relevant to each other, and it's job is forward that encodings to the next layer of the encoder.

The function of each encoder layer is to process its input to generate encodings, containing information about which parts of the inputs are relevant to each other. It passes its set of encodings to the next encoder layer as inputs.

If we take our project into picture, we would try to use the English words and then depending on the text, and what place the word is at, and what is it embedding can be used to eventually generate encodings those encodings will be forward to other encoding layer so that they layer could do any other encoding if necessary.

1.3 Decoder

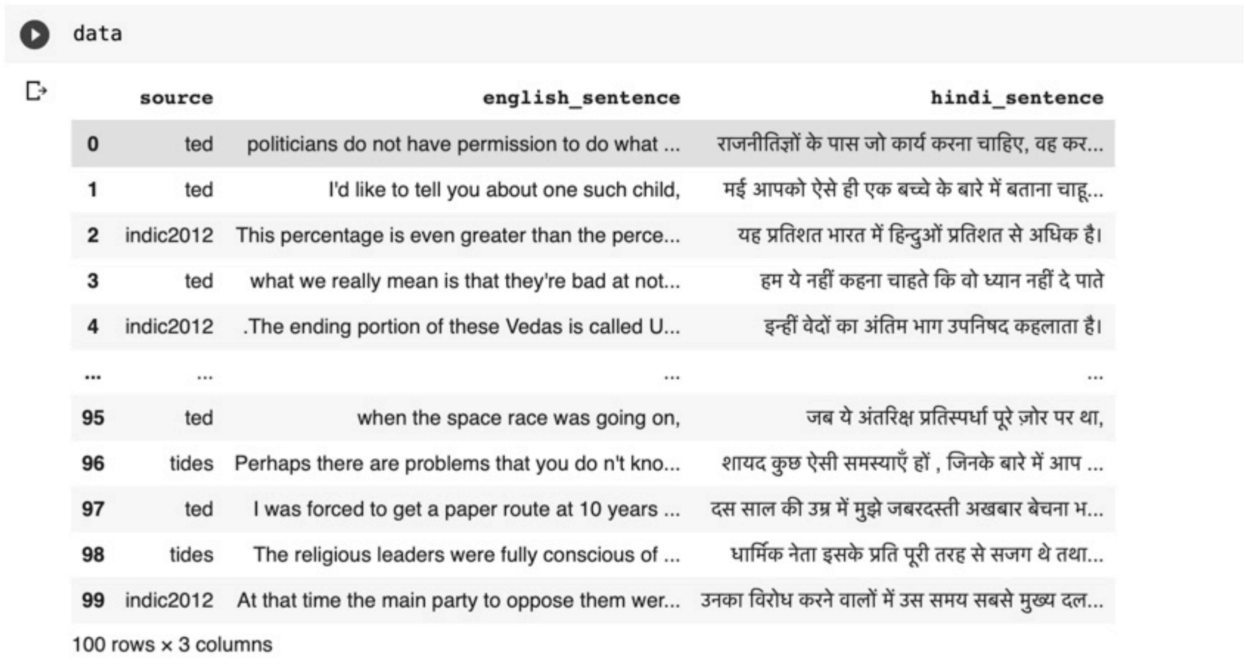
Decoder works just as opposite as the encoder, now as the encoder jobs is to use the word one and turn it into encoding, the job of the decoder would be just opposite, it will try to turn the words into its equivalent translated word.

So for example if we take, our task of the project, let's say we are translating india is a beautiful country, then the job of the decoder is to use the encodings of inda is a beautiful country, and use those encodings and with the help of that convert it into hindi ***भारत खूबसूरत देश है.***

The transformer is the combination of the encoder and decoder.

2 Dataset

HindEnCorp parallel texts (sentence-aligned) come from the following sources: Tides, which contains 50K sentence pairs taken mainly from news articles. This dataset was originally collected for the DARPA-TIDES surprise-language con-test in 2002, later refined at IIIT Hyderabad and provided for the NLP Tools Contest at ICON 2008 (Venkatapathy, 2008). This corpus (Baker et al., 2002) consists of three components: monolingual, parallel and annotated corpora. There are fourteen monolingual sub- corpora, including both written and (for some languages) spoken data for fourteen South Asian languages. The EMILLE monolingual corpora contain in total 92,799,000 words (including 2,627,000 words of transcribed spoken data for Bengali, Gujarati, Hindi, Punjabi and Urdu). The parallel corpus consists of 200,000 words of text in English and its accompanying translations into Hindi and other languages. the parallel corpus using these sources: Intercorp (Čermák and Rosen,2012) is a large multilingual parallel corpus of 32 languages including Hindi. The central language used for alignment is Czech. Intercorp's core texts amount to 202 million words. These core texts are most suitable for us because their sentence alignment is manually checked and therefore very reliable. They cover predominately short stories and novels. There are seven Hindi texts in Inter- corp. Unfortunately, only for three of them the English translation is available; the other four are aligned only with Czech texts. The Hindi subcorpus of Intercorp contains 118,000 words in Hindi. Here is the snippet of the dataset. The following is the screenshot of the dataset.



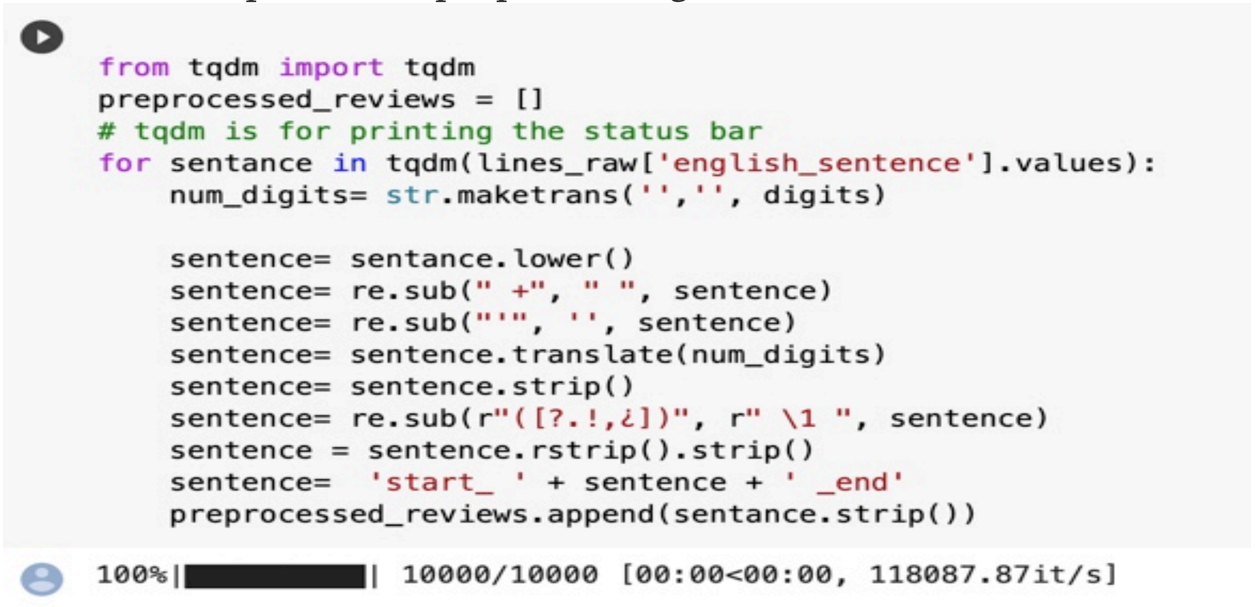
```
data
```

	source	english_sentence	hindi_sentence
0	ted	politicians do not have permission to do what ...	राजनीतिज्ञों के पास जो कार्य करना चाहिए, वह कर...
1	ted	I'd like to tell you about one such child,	मई आपको ऐसे ही एक बच्चे के बारे में बताना चाहूं...
2	indic2012	This percentage is even greater than the perce...	यह प्रतिशत भारत में हिन्दुओं प्रतिशत से अधिक है।
3	ted	what we really mean is that they're bad at not...	हम ये नहीं कहना चाहते कि वो ध्यान नहीं दे पाते
4	indic2012	.The ending portion of these Vedas is called U...	इन्हीं वेदों का अंतिम भाग उपनिषद कहलाता है।
...	...	...	...
95	ted	when the space race was going on,	जब ये अंतरिक्ष प्रतिस्पर्धा पूरे ज़ोर पर था,
96	tides	Perhaps there are problems that you do n't kno...	शायद कुछ ऐसी समस्याएँ हों , जिनके बारे में आप ...
97	ted	I was forced to get a paper route at 10 years ...	दस साल की उम्र में मुझे जबरदस्ती अखबार बेचना भ...
98	tides	The religious leaders were fully conscious of ...	धार्मिक नेता इसके प्रति पूरी तरह से सजग थे तथा...
99	indic2012	At that time the main party to oppose them wer...	उनका विरोध करने वालों में उस समय सबसे मुख्य दल...

100 rows × 3 columns

3. Preprocessing Techniques

In order for the data to be processed to the different models, it should be gone through different preprocessing techniques, such as cleaning, stemming, lemmatization, tokenization, padding, start and end symbols have been added, to the data. Following is the code and outputs after preprocessing of the data.

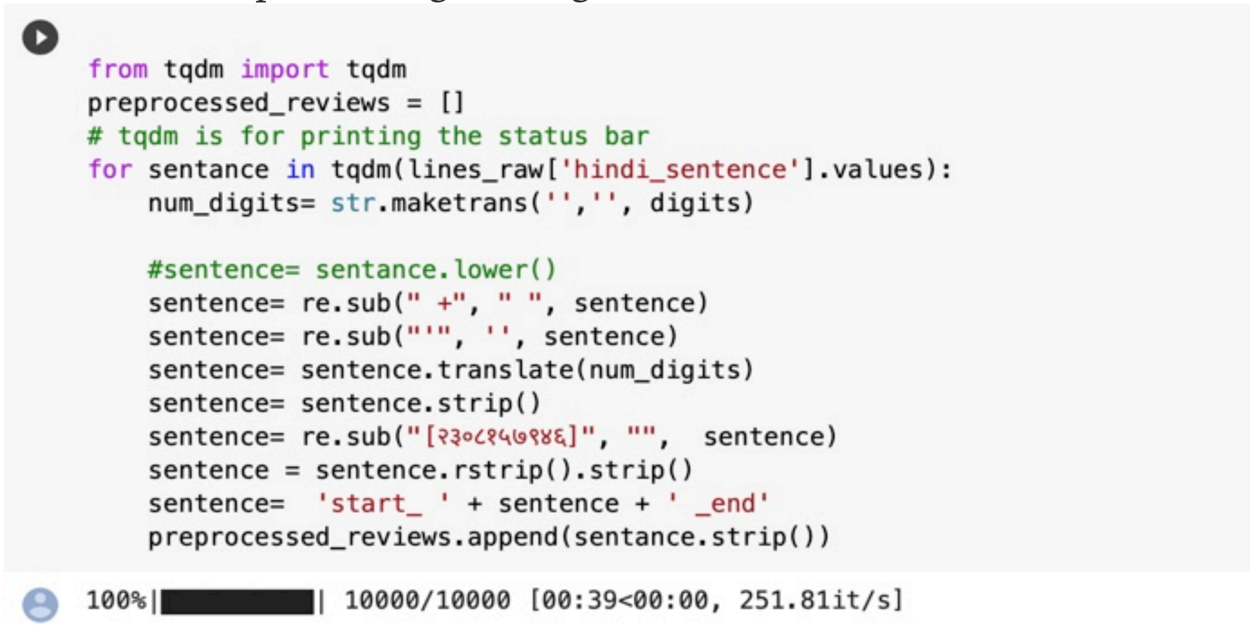


```
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(lines_raw['english_sentence'].values):
    num_digits= str.maketrans('','', digits)

    sentence= sentance.lower()
    sentence= re.sub(" +", " ", sentence)
    sentence= re.sub("'", '', sentence)
    sentence= sentence.translate(num_digits)
    sentence= sentence.strip()
    sentence= re.sub(r"([?.!,¿])", r" \1 ", sentence)
    sentence = sentence.rstrip().strip()
    sentence=  'start_ ' + sentence + ' _end'
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 10000/10000 [00:00<00:00, 118087.87it/s]
```

Preprocessing on English Sentences

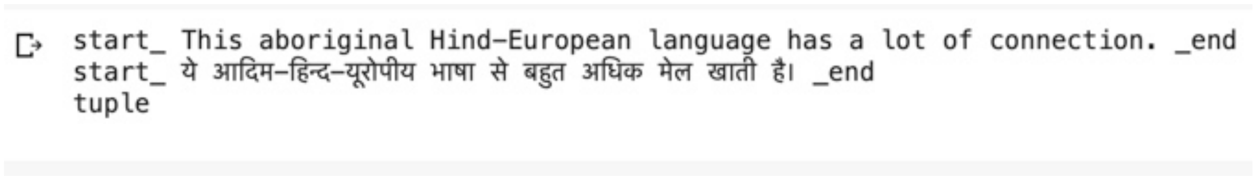


```
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(lines_raw['hindi_sentence'].values):
    num_digits= str.maketrans('','', digits)

    #sentence= sentance.lower()
    sentence= re.sub(" +", " ", sentence)
    sentence= re.sub("'", '', sentence)
    sentence= sentence.translate(num_digits)
    sentence= sentence.strip()
    sentence= re.sub("[२३०८१५७९४६]", "",  sentence)
    sentence = sentence.rstrip().strip()
    sentence=  'start_ ' + sentence + ' _end'
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 10000/10000 [00:39<00:00, 251.81it/s]
```

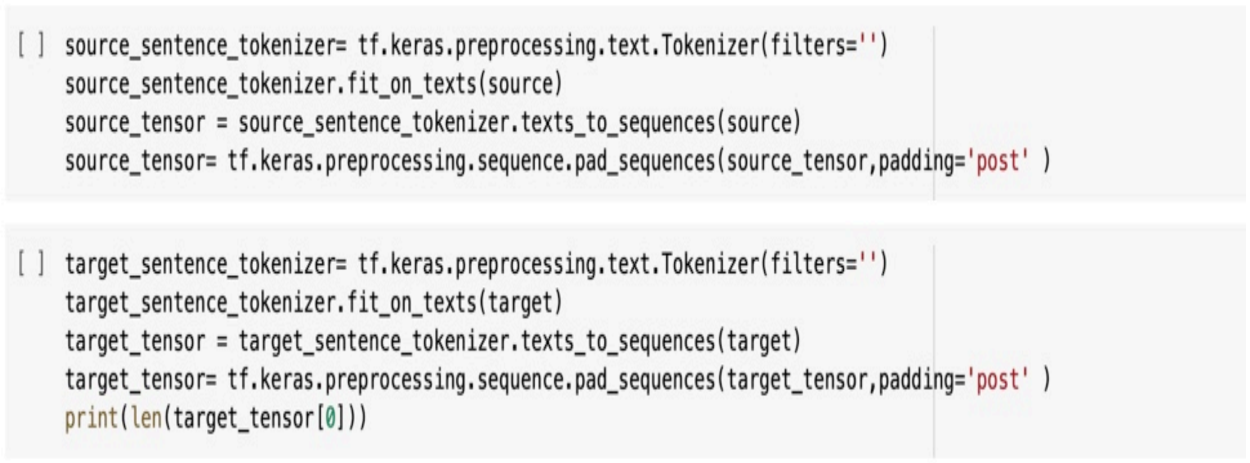
Preprocessing on Hindi Sentences



```
start_ This aboriginal Hind-European language has a lot of connection. _end
start_ ये आदिम-हिन्द-यूरोपीय भाषा से बहुत अधिक मेल खाती है। _end
tuple
```

Output after Preprocessing

Furthermore we need this text to be represented in numbers so we are converting the text into sequence with the help of keras and following is the code for the converting the text into sequence.



```
[ ] source_sentence_tokenizer= tf.keras.preprocessing.text.Tokenizer(filters='')
    source_sentence_tokenizer.fit_on_texts(source)
    source_tensor = source_sentence_tokenizer.texts_to_sequences(source)
    source_tensor= tf.keras.preprocessing.sequence.pad_sequences(source_tensor,padding='post' )
```

```
[ ] target_sentence_tokenizer= tf.keras.preprocessing.text.Tokenizer(filters='')
    target_sentence_tokenizer.fit_on_texts(target)
    target_tensor = target_sentence_tokenizer.texts_to_sequences(target)
    target_tensor= tf.keras.preprocessing.sequence.pad_sequences(target_tensor,padding='post' )
    print(len(target_tensor[0]))
```

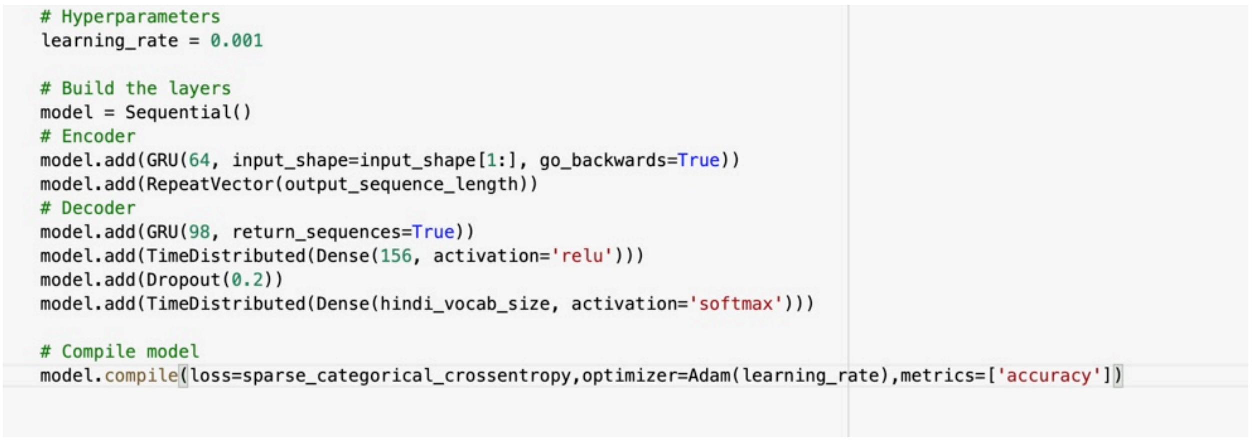
Converting text into sequence

After the dataset has been gone through the pre-processing phase, it will pass through the different models.

4. Models

1. Encoder & Decoder

Our first model consists of seq2seq model consists of GRU encoder and GRU decoder that are used for training the model, here in the words after the preprocessing is done, it's positional encoding is done, so that it can be converted into vector format. The GRU has a 3D output, now in order for the output process it was to be forward to the time series densed layer. Now after it passes thorough to the time series layer it has to be processed to the through the decoder (GRU) layer in between we will be using dropout layers for conversion. With the help of the translation output, and keras we find out the accuracy and loss of the output. A snippet of the code of the architecture implemented has been showed as follows.



```
# Hyperparameters
learning_rate = 0.001

# Build the layers
model = Sequential()
# Encoder
model.add(GRU(64, input_shape=input_shape[1:], go_backwards=True))
model.add(RepeatVector(output_sequence_length))
# Decoder
model.add(GRU(98, return_sequences=True))
model.add(TimeDistributed(Dense(156, activation='relu')))
model.add(Dropout(0.2))
model.add(TimeDistributed(Dense(hindi_vocab_size, activation='softmax')))

# Compile model
model.compile(loss=sparse_categorical_crossentropy,optimizer=Adam(learning_rate),metrics=['accuracy'])
```

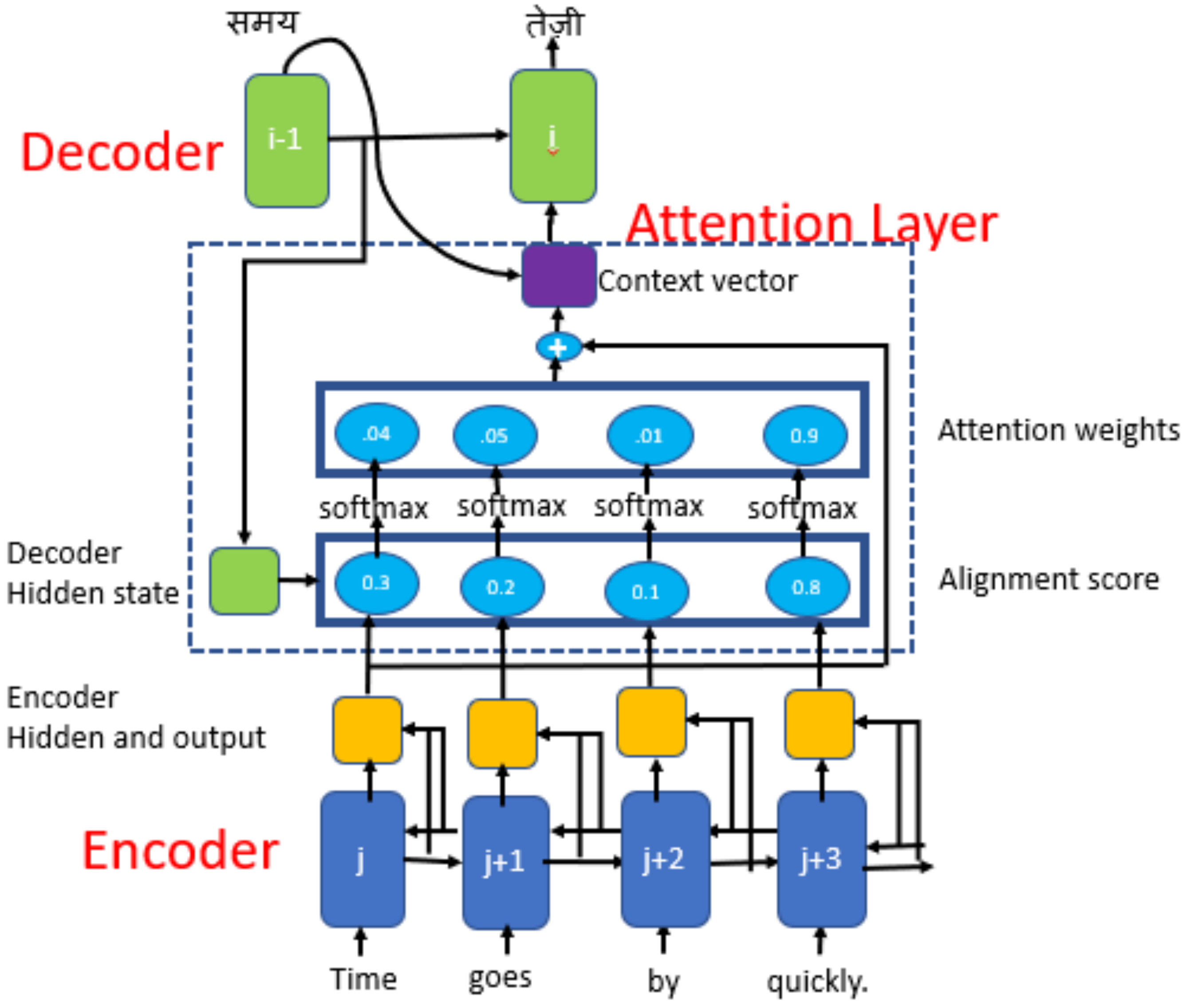
2 Sequence model with Attention Mechanism

Seq2Seq model maps a source sequence to the target sequence. The source sequence in the case of neural machine translation could be English, and the target sequence can be Hindi. We pass a source sentence in English to an encoder; the encoder encodes the complete information of the source sequence into a single real-valued vector, also known as the context vector. This context vector is then passed on the decoder to produce an output sequence in a target language like Hindi. The context vector has the responsibility to summarize the entire input sequence into a single vector. The basic idea of Attention mechanism is to avoid attempting to learn a single vector representation for each sentence, instead, it pays attention to specific input vectors of the input sequence based on the attention weights. At every decoding step, the decoder will be informed how much "attention" needs to be paid to each input word using a set of ***attention weights***. These attention weights provide contextual information to the decoder for translation.

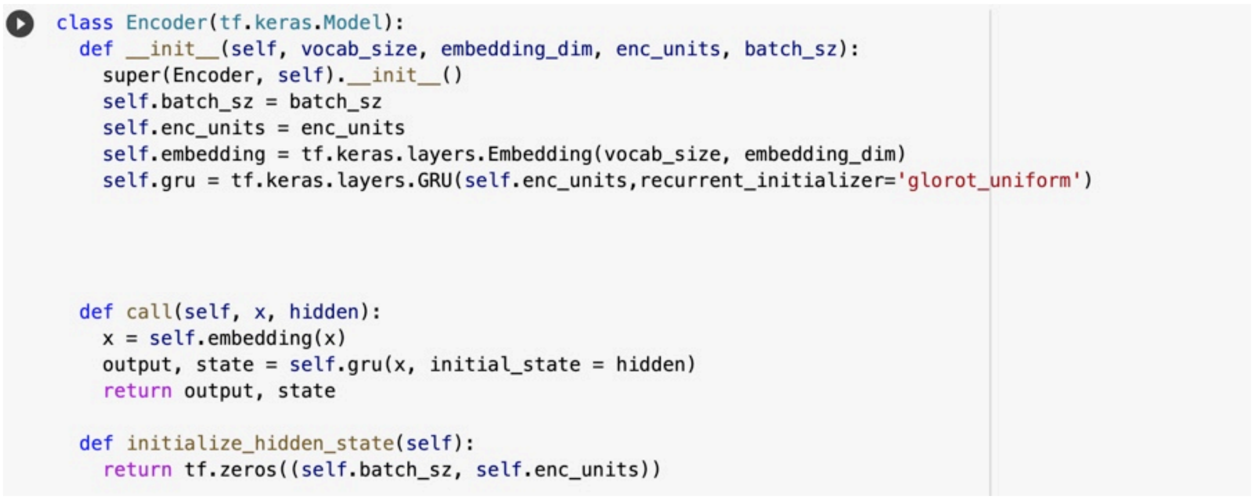
Bahdanau attention mechanism

Bahdanau et al. proposed an attention mechanism that learns to align and translate jointly. It is also known as Additive attention as it performs a linear combination of encoder states and the decoder states.
All hidden states of the encoder(forward and backward) and the decoder are used to generate the context vector, unlike how just the last encoder hidden state is used in seq2seq without attention.
The attention mechanism aligns the input and output sequences,

with an alignment score parameterized by a feed-forward network. It helps to pay attention to the most relevant information in the source sequence. The model predicts a target word based on the context vectors associated with the source position and the previously generated target words.



For our model which we have the seq2seq with attention we have taken the help of the above architecture and following is snippet of the code. The first snippet is about the encoder and the other architecture following it, is about the decoder.



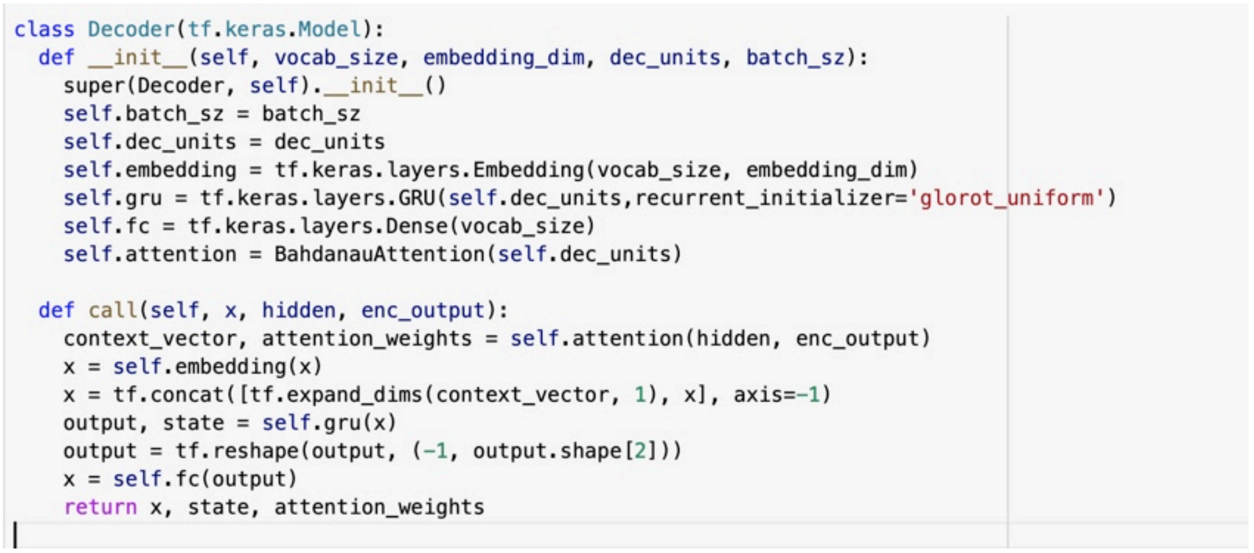
```python
class Encoder(tf.keras.Model):
  def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
    super(Encoder, self).__init__()
    self.batch_sz = batch_sz
    self.enc_units = enc_units
    self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
    self.gru = tf.keras.layers.GRU(self.enc_units,recurrent_initializer='glorot_uni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



  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))
```

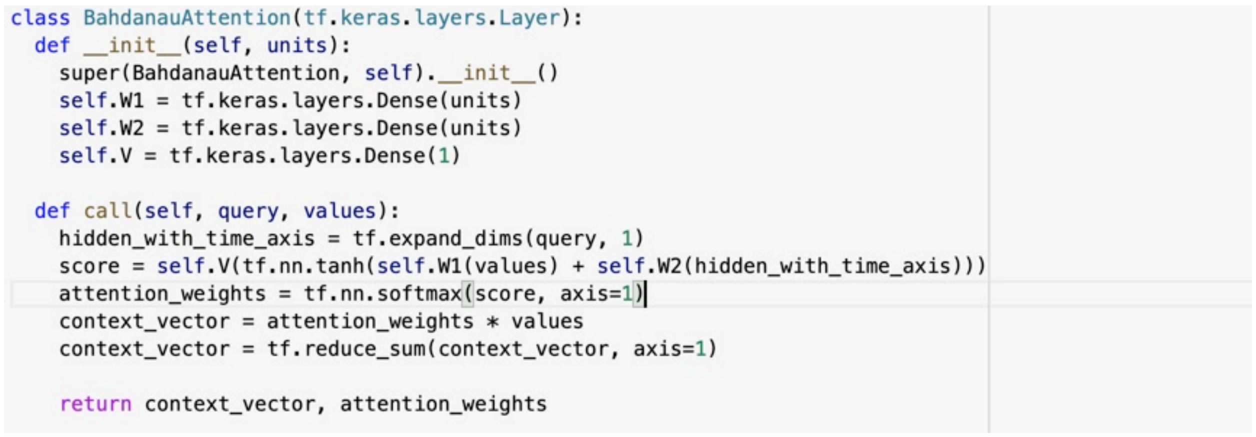
Code for encoder architecture.



```python
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,recurrent_initializer='glorot_uniform')
    self.fc = tf.keras.layers.Dense(vocab_size)
    self.attention = BahdanauAttention(self.dec_units)

  def call(self, x, hidden, enc_output):
    context_vector, attention_weights = self.attention(hidden, enc_output)
    x = self.embedding(x)
    x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)
    output, state = self.gru(x)
    output = tf.reshape(output, (-1, output.shape[2]))
    x = self.fc(output)
    return x, state, attention_weights
```

Code for decoder architecture



```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
```

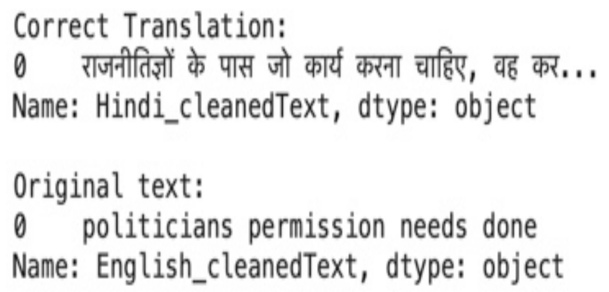
Code for attention architecture

5. Evaluation

Now from the two models we try to find out which performs better, which could we used for better machine translation, for that we need some metrics to calculate the evaluation. is the Evaluation of all the models has been done with different parameters such as losses, and accuracy. And for comparing how good the machine is performing we can analyse the output and with some metrics. We try to measure both of the models performance with evaluation metrics. We have use hyper-parameters such as learning rate for the arriving to the solution quickly. We have values such as accuracy and losses as our evaluation metrics. Following is the screenshot of both the outputs.

Model	**Accuracy**	**Losses**
Encoder and Decoder	91.4	0.79
Encoder and Decoder with attention	92.3	0.695

After calculating both the losses and accuracy we will have the final output, and the output of the machine translation is as follows.



```
Correct Translation:
0    राजनीतिज्ञों के पास जो कार्य करना चाहिए, वह कर...
Name: Hindi_cleanedText, dtype: object

Original text:
0    politicians permission needs done
Name: English_cleanedText, dtype: object
```

6. Conclusion

After performing the experiments, and evaluating the experiments on the models, both models are able to translate the text from English to Hindi, the model with the bahdanau attention performs a bit better with low loss and better accuracy. There are few models yet to be explored and we can come with better models and logic and perform even better models and architecture and find better translator. Now if we collect more data then we can find a way machine translation could be done on different accents, as a language can have multiple accents.

7. References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Britz, D., Goldie, A., Luong, M., and Le, Q. V. (2017). Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.