<u>Text Analytics Case Study -</u>

Word Level English to Marathi Neural Machine Translation using Encoder-Decoder Model

Submitted By:

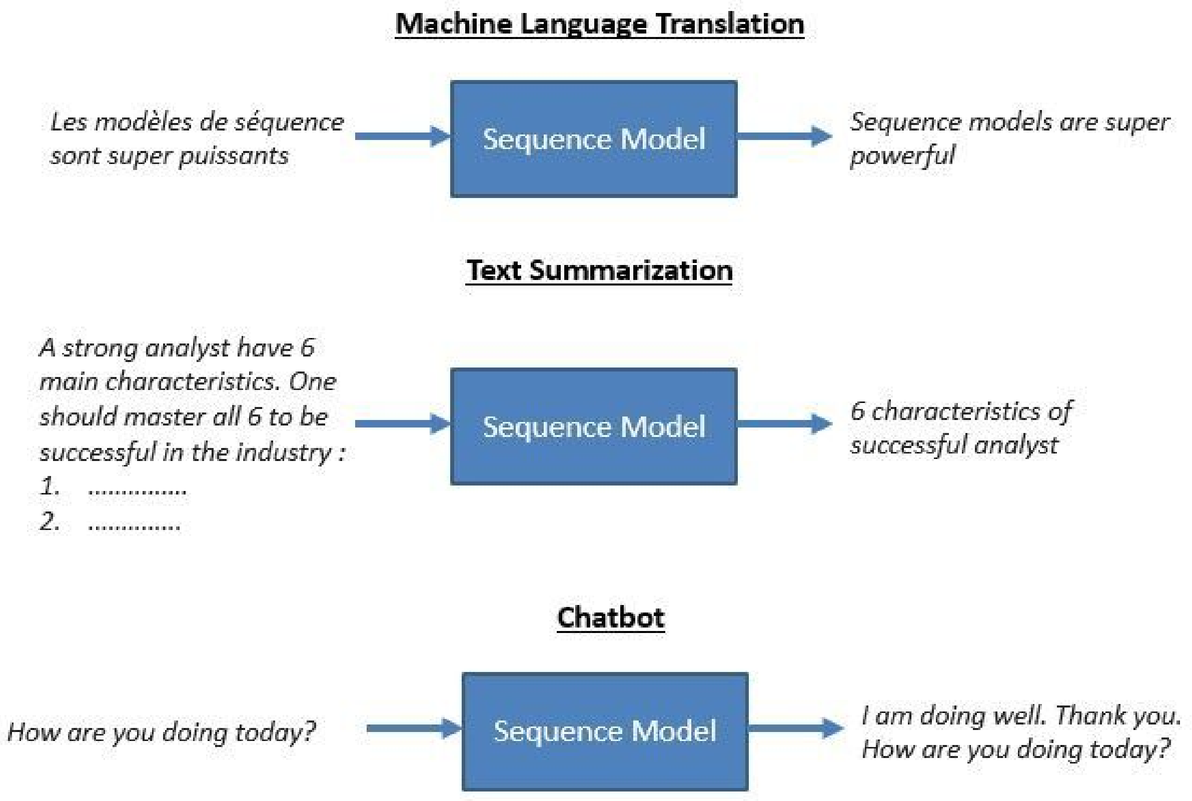
Anoushka Motwani (20)

Devanshi Sukhija (46)

Table of Contents:

Introduction

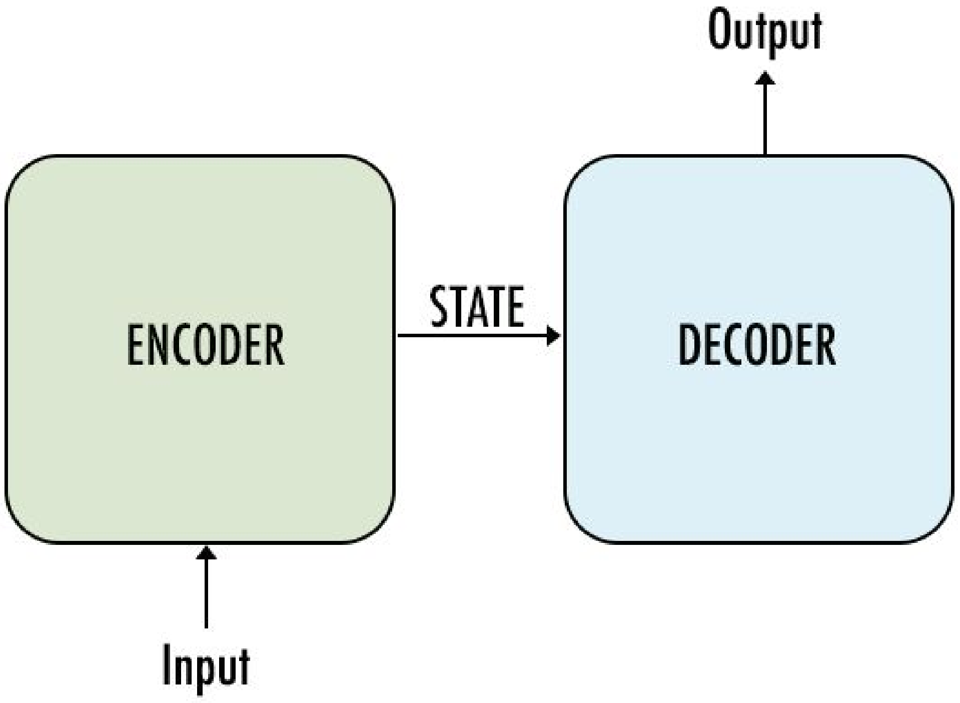
Recurrent Neural Networks (RNN) have real time applications in speech recognition, natural Language Processing (NLP) problems.
Sequence to Sequence (seq2seq) models are a special class of RNN architectures usually used to solve complex language related problems like Machine translation, Question Answering, creating Chat-bots and Text Summarization.


Machine Language Translation
Les modèles de séquence sont super puissants
Sequence Model
Sequence models are super powerful
Text Summarization
A strong analyst have 6 main characteristics. One should master all 6 to be successful in the industry :
1. ....
2. ....
Sequence Model
6 characteristics of successful analyst
Chatbot
How are you doing today?
Sequence Model
I am doing well. Thank you. How are you doing today?

In this case study we are focusing on seq2seq models, how they are built and how they solve complex tasks. Here, we are taking a problem Machine Translation.
We are translating a text from English to Marathi.

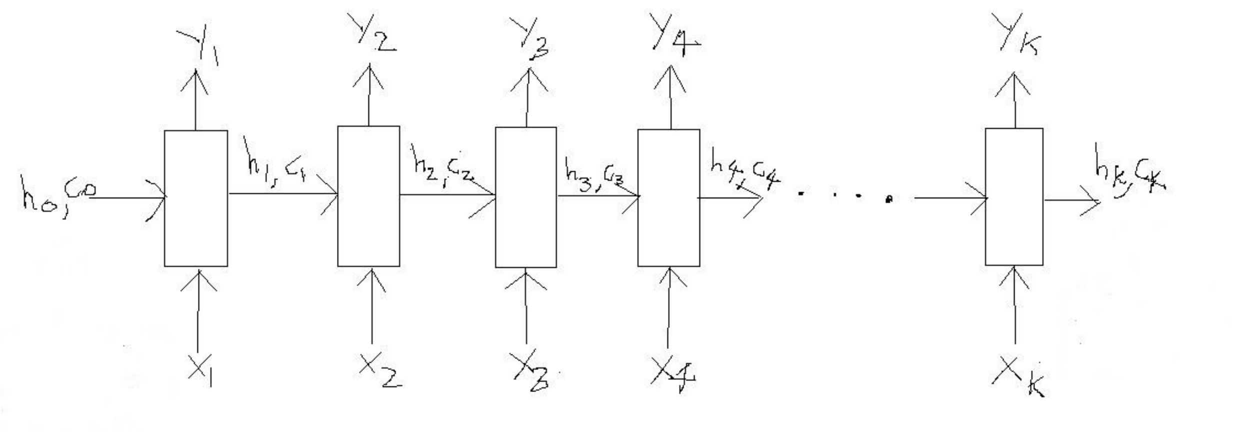
Encoder - Decoder Architecture

The Encoder Decoder architecture is the most common architecture that is used to build seq2seq models.



- Encoder and Decoder are both typically LSTM models, or sometimes GRU models.
- The Encoder reads the input sequence and summarizes the information in internal state vectors or hidden state and cell state vectors. We discard the outputs of the encoder and only preserve the internal states.
- The Decoder is an LSTM whose initial states are initialized to the final states of the Encoder LSTM. Using these initial states, Decoder starts generating the output sequence.
- The decoder behaves a bit differently during the training and inference procedure. During the training, we use a technique call teacher forcing which helps to train the decoder faster. During inference, the input to the decoder at each time step is the output from the previous time step.
- Intuitively, the encoder summarizes the input sequence into state vectors or thought vectors, which are then fed to the decoder which starts generating the output sequence given the Thought vectors. The decoder is just a language model conditioned on the initial states.

Encoder LSTM



The LSTM reads the data one sequence after the other. Thus if the input is a sequence of length ‘k’, we say that LSTM reads it in ‘k’ time steps.
With respect to the above diagram, following are the 3 main components of an LSTM:

1. Xi => Input sequence at time step i
2. hi and ci => LSTM maintains two states (‘h’ for hidden state and ‘c’ for cell state) at each time step. Combined together these are internal state of the LSTM at time step i.
3. Yi => Output sequence at time step i.

Example:

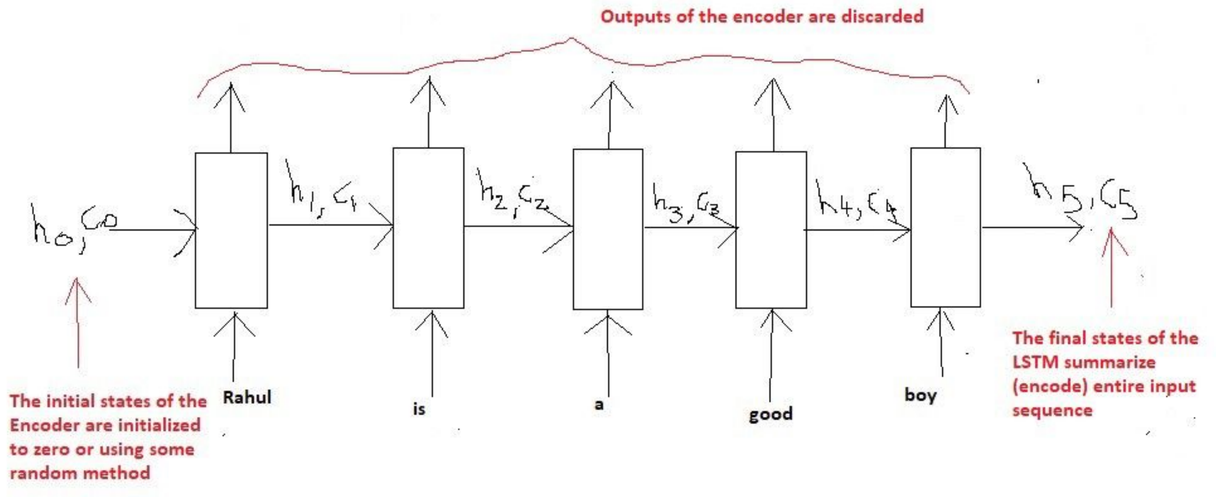
Input sentence (English)=> “Rahul is a good boy”
Output sentence (Marathi) => “राहुल चांगला मुलगा आहे”

Explanation for Xi:

A sentence can be seen as a sequence of either words or characters. For example in the case of words, the above English sentence can be thought of as a sequence of 5 words (‘Rahul’, ‘is’, ‘a’, ‘good’, ‘boy’). And in the case of characters, it can be thought of as a sequence of 19 characters (‘R’, ‘a’, ‘h’, ‘u’, ‘l’, ‘ ‘, ……, ‘y’).

We will break the sentence by words as this scheme is more common in real world applications. Hence the name ‘Word Level NMT’. So, referring to the diagram above, we have the following input:
X1 = ‘Rahul’, X2 = ‘is’, X3 = ‘a’, X4 = ‘good, X5 = ‘boy’.

The LSTM will read this sentence word by word in 5 time steps as follows:


Outputs of the encoder are discarded
h0, c0
h1, c1
h2, c2
h3, c3
h4, c4
h5, c5
Rahul
is
a
good
boy
The initial states of the Encoder are initialized to zero or using some random method
The final states of the LSTM summarize (encode) entire input sequence

Explanation for hi and ci:

The internal states, hi and ci, remember what the LSTM has learned till now.
Example:
h3, c3 =>These two vectors will remember that the network has read “Rahul is a” till now. Basically it’s the summary of information till time step 3 which is stored in the vectors h3 and c3 (thus called the states at time step 3).
Similarly, we can thus say that h5, c5 will contain a summary of the entire input sentence, since this is where the sentence ends (at time step 5). These states coming out of the last time step are also called as the “Thought vectors” as they summarize the entire sequence in a vector form.

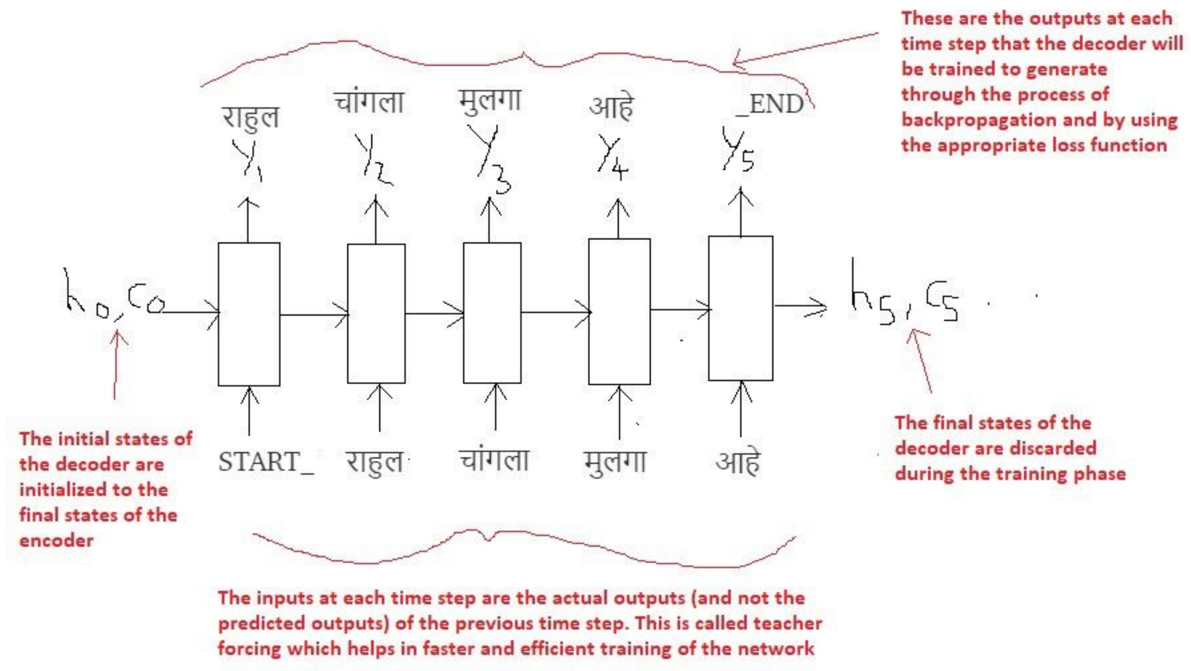
Explanation for Yi:

These are the predictions of the LSTM model at each time step. In the case of word level language models each Yi is actually a probability distribution over the entire vocabulary which is generated by using a softmax activation. Thus each Yi is a vector of size “vocab_size” representing a probability distribution. Depending on the context of the problem they might sometimes be used or sometimes be discarded.
In our case we have nothing to output unless we have read the entire English sentence. Because we will start generating the output sequence (equivalent Marathi sentence) once we have read the entire English sentence. Thus we will discard the Yi of the Encoder for our problem.

Decoder LSTM - Training Model

Unlike the Encoder LSTM which has the same role to play in both the training phase as well as in the inference phase, the Decoder LSTM has a slightly different role to play in both of these phases.
The given input sentence is “Rahul is a good boy”, the goal of the training process is to train (teach) the decoder to output “राहुल चांगला मुलगा आहे”. Just as the Encoder scanned the input sequence word by word, similarly the Decoder will generate the output sequence word by word.

We add two tokens in the output sequence as follows:
Output sequence => “START_ राहुल चांगला मुलगा आहे _END”


These are the outputs at each time step that the decoder will be trained to generate through the process of backpropagation and by using the appropriate loss function
राहुल
चांगला
मुलगा
आहे
_END
Y_1
Y_2
Y_3
Y_4
Y_5
h_0, c_0
h_5, c_5
START_
राहुल
चांगला
मुलगा
आहे
The initial states of the decoder are initialized to the final states of the encoder
The final states of the decoder are discarded during the training phase
The inputs at each time step are the actual outputs (and not the predicted outputs) of the previous time step. This is called teacher forcing which helps in faster and efficient training of the network

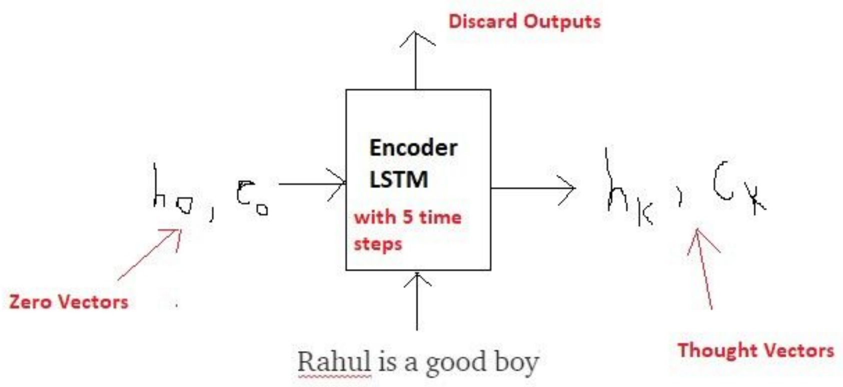
The most important point is that the initial states (h0, c0) of the decoder are set to the final states of the encoder. This intuitively means that the decoder is trained to start generating the output sequence depending on the information encoded by the encoder. Obviously the translated Marathi sentence must depend on the given English sentences.
We provide the START_ token so that the decoder starts generating the next token (the actual first word of Marathi sentence). And after the last word in the Marathi sentence, we make the decoder learn to predict the _END token. This will be used as the stopping condition during the inference procedure, basically it will denote the end of the translated sentence and we will stop the inference loop.

We use a technique called “Teacher Forcing” wherein the input at each time step is given as the actual output (and not the predicted output) from the previous time step. This helps in faster and efficient training of the network.
Finally the loss is calculated on the predicted outputs from each time step and the errors are back propagated through time in order to update the parameters of the network. Training the network over longer period with sufficiently large amount of data results in pretty good predictions.

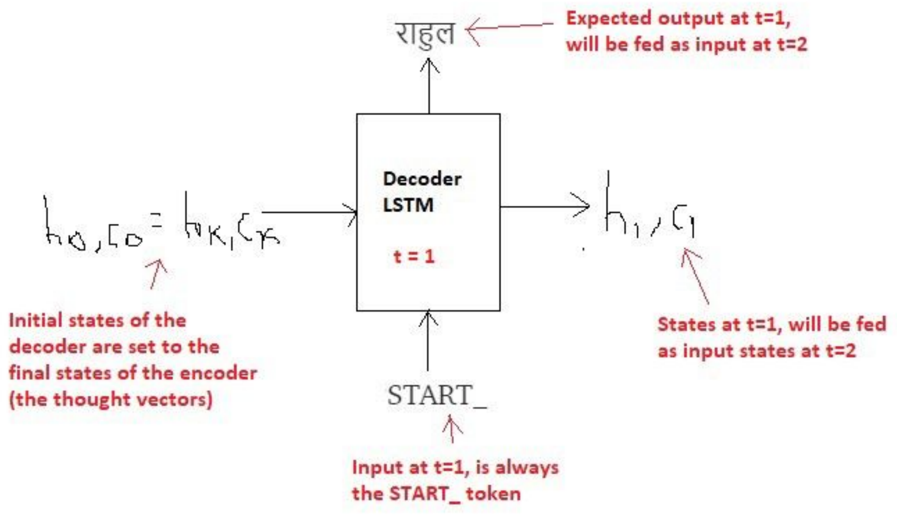
Decoder LSTM — Inference Mode

The decoder now has to predict the entire output sequence (Marathi sentence) given these thought vectors.
Input sequence => “Rahul is a good boy”
(Expected) Output Sequence => “राहुल चांगला मुलगा आहे”

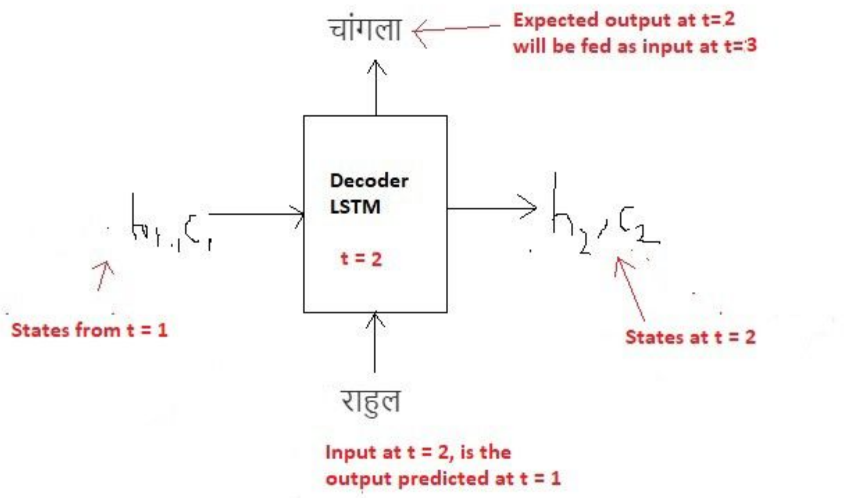
Step 1: Encode the input sequence into the Thought Vectors



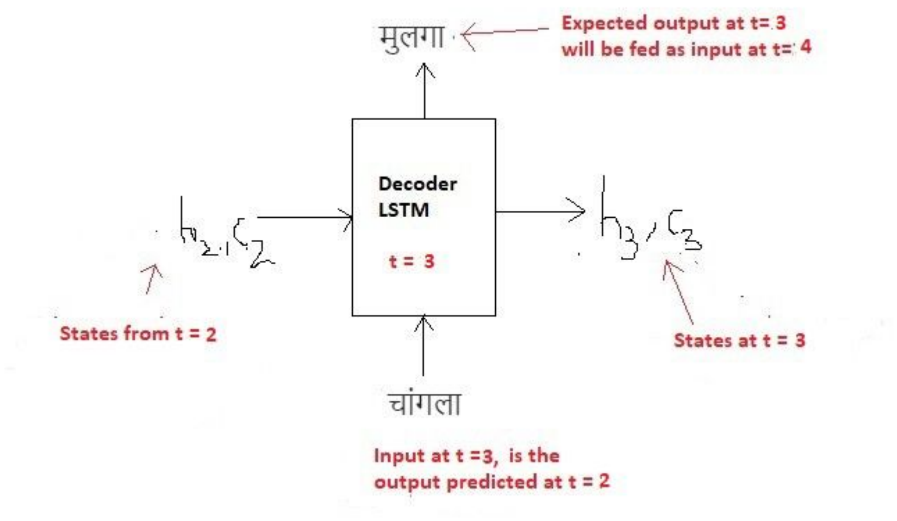
Step 2: Start generating the output sequence in a loop, word by word.
At T=1


राहुल
Expected output at t=1, will be fed as input at t=2
Decoder LSTM
t = 1
h0,c0 = hk,ck
h1,c1
Initial states of the decoder are set to the final states of the encoder (the thought vectors)
States at t=1, will be fed as input states at t=2
START_
Input at t=1, is always the START_ token

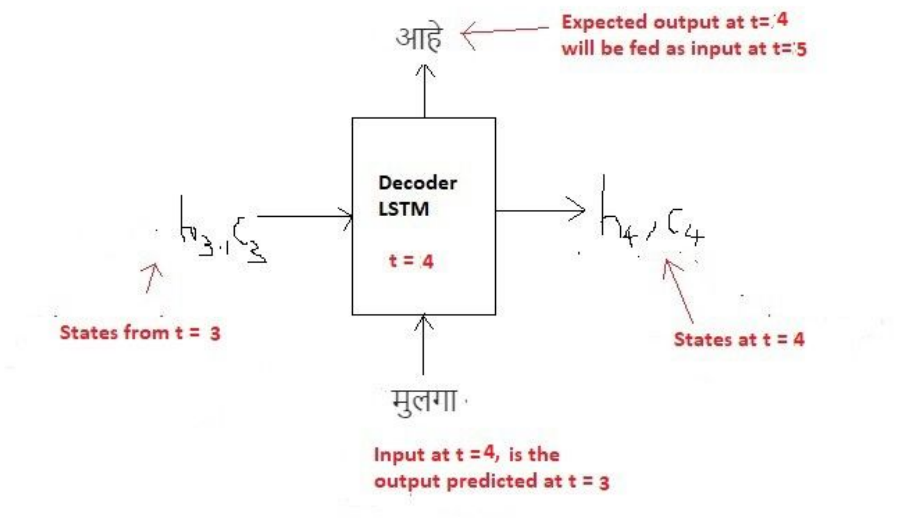
At T=2


चांगला
Expected output at t=2 will be fed as input at t=3
Decoder LSTM
t = 2
h1,c1
h2,c2
States from t = 1
States at t = 2
राहुल
Input at t = 2, is the output predicted at t = 1

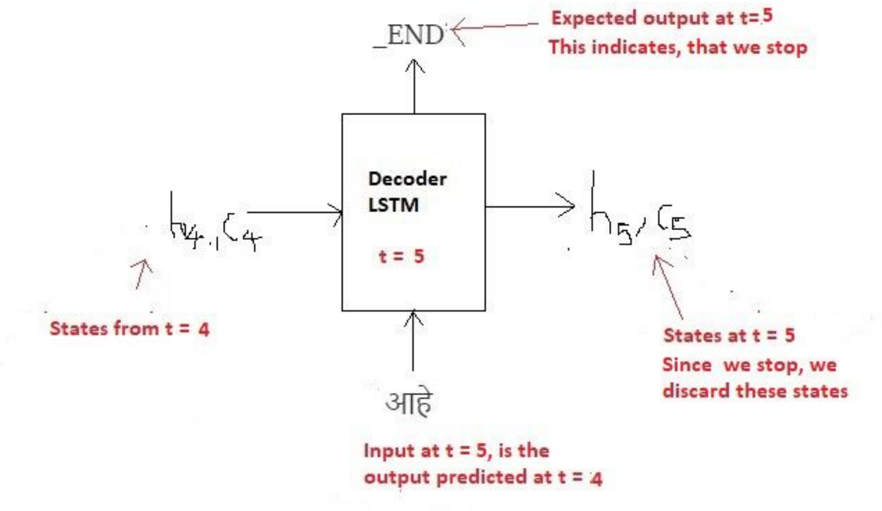
At T=3


मुलगा
Expected output at t= 3
will be fed as input at t= 4
Decoder
LSTM
t = 3
h_2, c_2
States from t = 2
h_3, c_3
States at t = 3
चांगला
Input at t =3, is the
output predicted at t = 2

At T=4


आहे
Expected output at t= 4
will be fed as input at t= 5
Decoder
LSTM
t = 4
h_3, c_3
States from t = 3
h_4, c_4
States at t = 4
मुलगा
Input at t =4, is the
output predicted at t = 3

At T=5


_END
Expected output at t=5
This indicates, that we stop
Decoder LSTM
t = 5
h4, c4
States from t = 4
h5, c5
States at t = 5
Since we stop, we discard these states
आहे
Input at t = 5, is the output predicted at t = 4

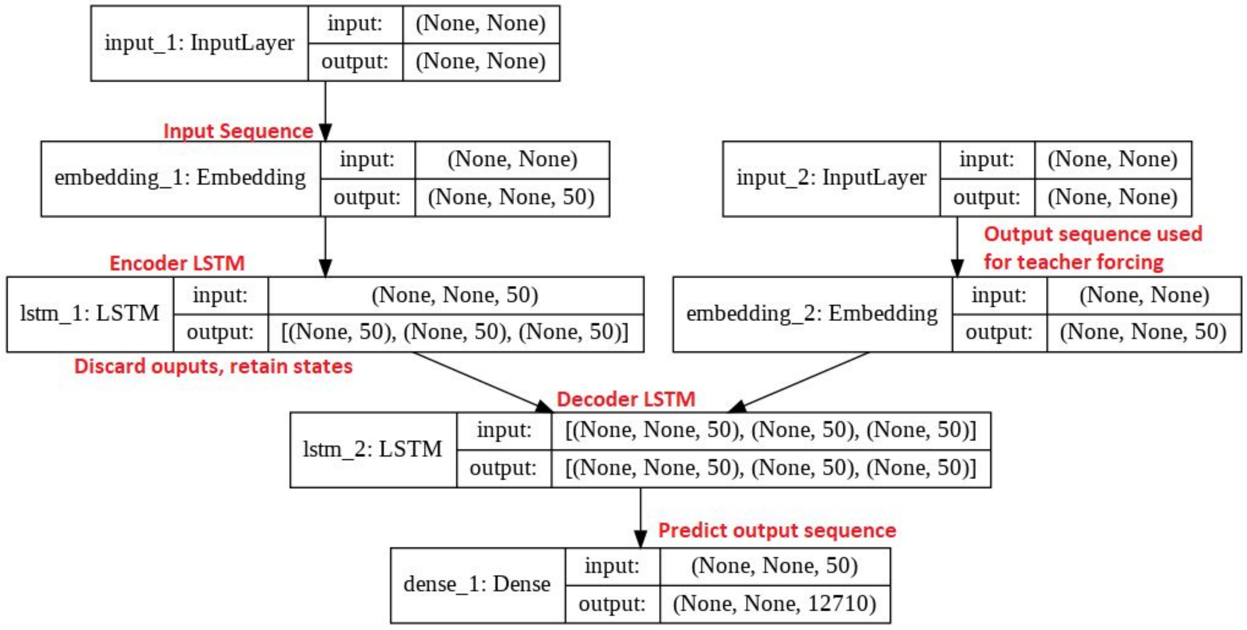
Inference Algorithm:

1. During inference, we generate one word at a time. Thus the Decoder LSTM is called in a loop, every time processing only one time step.
2. The initial states of the decoder are set to the final states of the encoder.
3. The initial input to the decoder is always the START_ token.
4. At each time step, we preserve the states of the decoder and set them as initial states for the next time step.
5. At each time step, the predicted output is fed as input in the next time step.
6. We break the loop when the decoder predicts the END_ token.

Code Walk-Through

We picked up our dataset from http://www.manythings.org/anki/
First we performed data cleaning and preparation. We then compute the vocabulary for both English and Marathi, we also calculated the vocabulary sizes and the length of maximum sequence for both the languages, finally creating 4 python dictionaries (2 for each language) to convert a given token into an integer index and vice-versa.
In the next step we make a 90-100 train and test split and write a Python generator function to load the data in batches after which we define the modell required for training.

Model Architecture:


input_1: InputLayer | input: (None, None) | output: (None, None)
Input Sequence
embedding_1: Embedding | input: (None, None) | output: (None, None, 50)
input_2: InputLayer | input: (None, None) | output: (None, None)
Output sequence used for teacher forcing
Encoder LSTM
lstm_1: LSTM | input: (None, None, 50) | output: [(None, 50), (None, 50), (None, 50)]
embedding_2: Embedding | input: (None, None) | output: (None, None, 50)
Discard ouputs, retain states
Decoder LSTM
lstm_2: LSTM | input: [(None, None, 50), (None, 50), (None, 50)] | output: [(None, None, 50), (None, 50), (None, 50)]
Predict output sequence
dense_1: Dense | input: (None, None, 50) | output: (None, None, 12710)

We train the network for 50 epochs with a batch size of 128.
Finally we generate the output sequence by invoking the setup.

Results and Evaluation

The purpose of this case study was to give an intuitive explanation on how to build sequence to sequence models using LSTM, the results are not world class for many reasons. The most important reason being is that the dataset size is very small, only 33000 pairs of sentences.

Evaluation on train dataset:

Input English sentence: it is a holiday tomorrow
Actual Marathi Translation: उद्या सुट्टी आहे
Predicted Marathi Translation: उद्या नाताळ आहे

Input English sentence: i will give you this book
Actual Marathi Translation: मी तुम्हाला हे पुस्तक देईन
Predicted Marathi Translation: मी तुला हे पुस्तक तुला दिलं

Input English sentence: this sentence has five words
Actual Marathi Translation: या वाक्यात पाच शब्द आहेत
Predicted Marathi Translation: या पाच शब्द आहेत

Input English sentence: did you clean your room
Actual Marathi Translation: तुझी खोली साफ केलीस का
Predicted Marathi Translation: तुझी खोली साफ केली का

Input English sentence: she wrapped herself in a blanket
Actual Marathi Translation: त्यांनी स्वतःला एका चादरीत गुंडाळून घेतलं
Predicted Marathi Translation: तिने एका छोट्या चादर घातली

Input English sentence: he is still alive
Actual Marathi Translation: तो अजूनही जिवंत आहे
Predicted Marathi Translation: ते अजूनही जिवंत आहेत

Input English sentence: you cant speak french can you
Actual Marathi Translation: तुला फ्रेंच बोलता येत नाही ना
Predicted Marathi Translation: तुला फ्रेंच बोलता येत नाही का नाही माहीत

Evaluation on test dataset:

Input English sentence: my wife isnt beautiful yours is
Actual Marathi Translation: माझी बायको सुंदर नाहीये तुझी आहे
Predicted Marathi Translation: माझी तुझी गर्लफ्रेंड सुंदर आहे भेटलो

Input English sentence: who lives in this house
Actual Marathi Translation: या घरात कोण राहतं
Predicted Marathi Translation: या घरात कोणी असतो

Input English sentence: somethings happened to tom
Actual Marathi Translation: टॉमला काहीतरी झालंय
Predicted Marathi Translation: टॉमला काहीतरी झालं आहे

Input English sentence: the dog jumped over a chair
Actual Marathi Translation: कुत्र्याने खुर्चीवरून उडी मारली
Predicted Marathi Translation: एक कुत्र्याला दरवाजा बंद केला

Input English sentence: i can prove who the murderer is
Actual Marathi Translation: खुनी कोण आहे हे मी सिद्ध करू शकतो
Predicted Marathi Translation: मी जिंकू शकतो हे मला माहीत आहे

Input English sentence: everyone could hear what tom said
Actual Marathi Translation: टॉम जे म्हणाला ते सर्वांना ऐकू येत होतं
Predicted Marathi Translation: टॉमने काय म्हटलं म्हटलं तर खरं

Input English sentence: those are my books
Actual Marathi Translation: ती माझी पुस्तकं आहेत
Predicted Marathi Translation: ती माझी पुस्तकं आहेत

Conclusion

Even though the results are not the best, they are not that bad as well. Certainly much better than what a randomly generated sequence would result in. In some sentences we can even note that the words predicted are not correct but they are semantically quite close to the correct words.
Also, another point to be noticed is that the results on training set are a bit better than the results on test set, which indicates that the model might be over-fitting a bit.

Future Work

To improve the quality the following measures can be taken:

1. Get much more data. Top quality translators are trained on millions of sentence pairs.
2. Build more complex models like Attention.
3. Use dropout and other forms of regularization techniques to mitigate over-fitting.
4. Perform Hyper-parameter tuning. Play with learning rate, batch size, dropout rate, etc. Try using bidirectional Encoder LSTM. Try using multi-layered LSTMs.
5. Try using beam search instead of a greedy approach.
6. Try BLEU score to evaluate your model.
7. The list is never ending and goes on.

References

1. https://medium.com/@dev.elect.iitd/neural-machine-translation-using-word-level-seq2seq-model-47538cba8cd7
2. https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
3. https://arxiv.org/abs/1409.3215
4. https://arxiv.org/abs/1406.1078