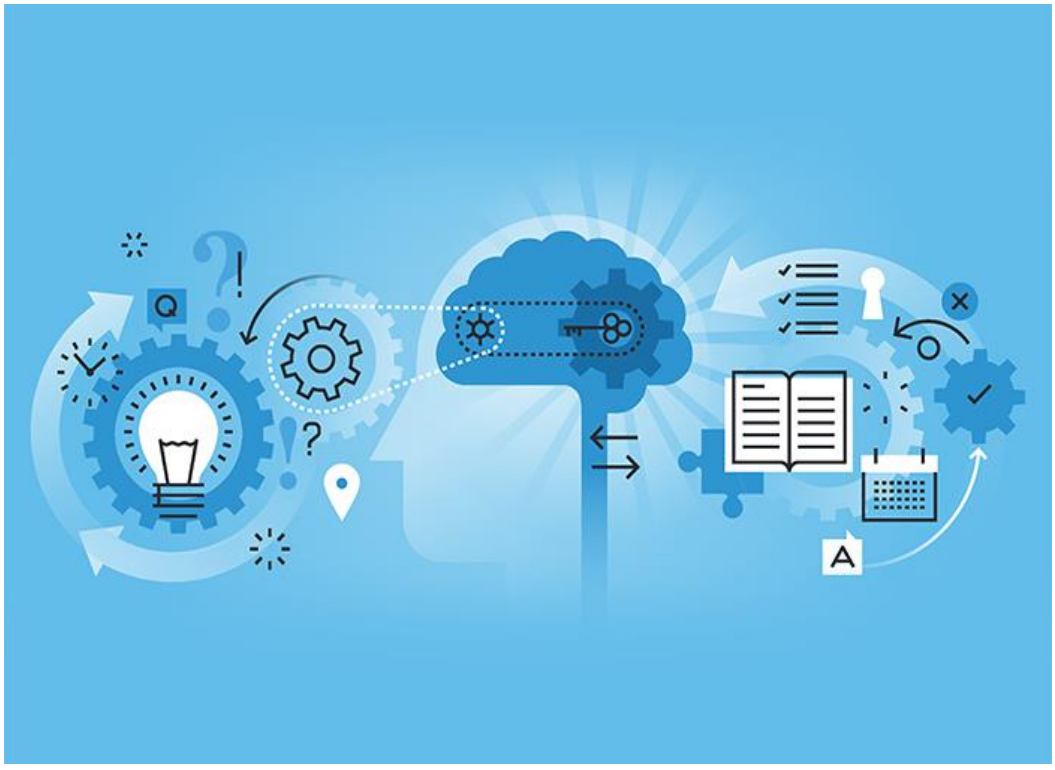
Indian Institute of Information Technology, Allahabad

Neural Machine Translation



Submitted by:

Ajay Kumar Singh

Project Guide:

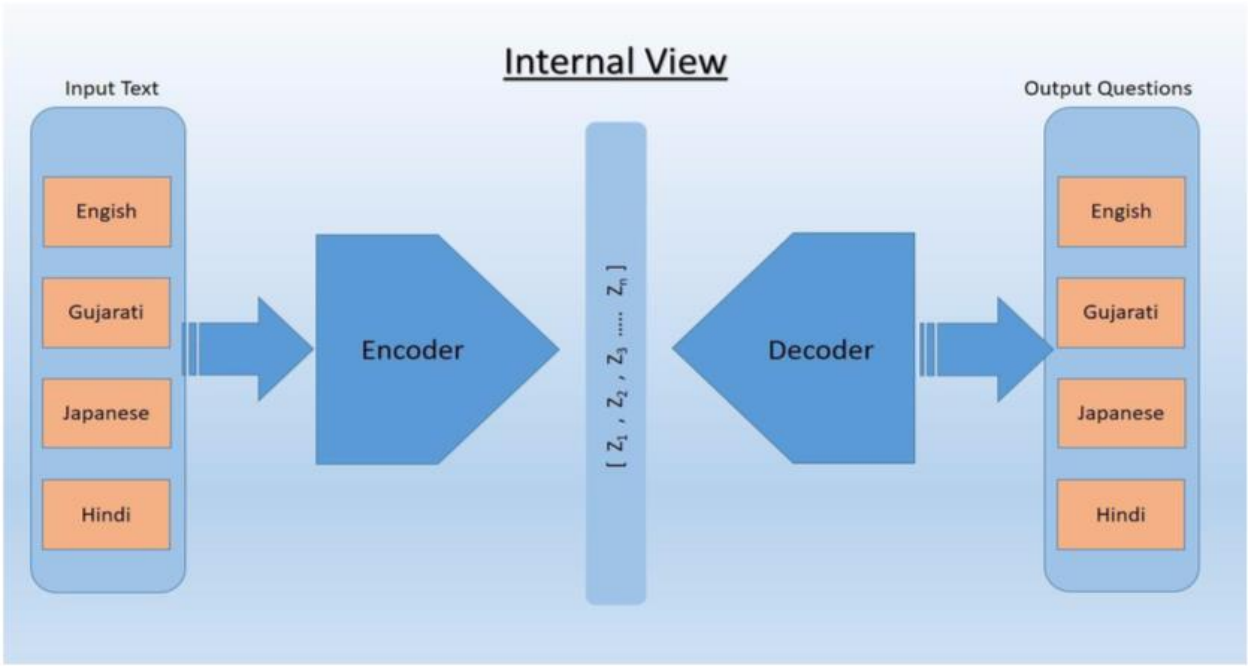
Dr. K.P Singh

INDICE

Machine Translation

Neural Machine Translation (NMT) is an end-to-end learning approach for automated translation, with the potential to overcome many of the weaknesses of conventional phrase-based translation systems. Deep neural networks can achieve excellent results in very complicated tasks (speech/visual object recognition), but despite their flexibility, they can be applied only for tasks where the input and target have fixed dimensionality. This issue is known as Deep Learning translation problem and it can be partially solved with the help of Recurrent Neural Network (RNN).

The basic principles of machine translation engines



Google Machine Translation

2. Algorithm Details – How it does?

In 2016, Google introduced the Encoder-Decoder neural network architecture for the language translation which outperforms all other language translation tools till this date. This architecture is also known as Sequence to Sequence models. Here is the overview of this architecture.

There are two parts of the sequence to sequence model. They are as follows:

2.1 Encoder

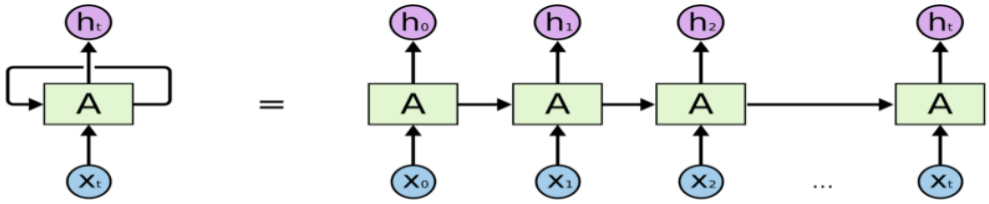
The encoder takes a pre-processed data from the input text and converts it according to the weights of the hidden layer. This hidden layer creates an intermediate representation of the input text and passes it to the decoder.

2.2 Decoder

The decoder takes this information from the hidden layer and converts it into another language. It produces the output in any desired language.

2.3 Long Short-Term Memory networks (LSTMs) and Recurrent Neural Network (RNN)

Here is where Long Short-Term Memory networks (LSTMs) come into play, helping us to work with sequences whose length we can't know a priori. LSTMs are a special kind of recurrent neural network (RNN), capable of learning long-term dependencies. All RNNs look like a chain of repeating modules.

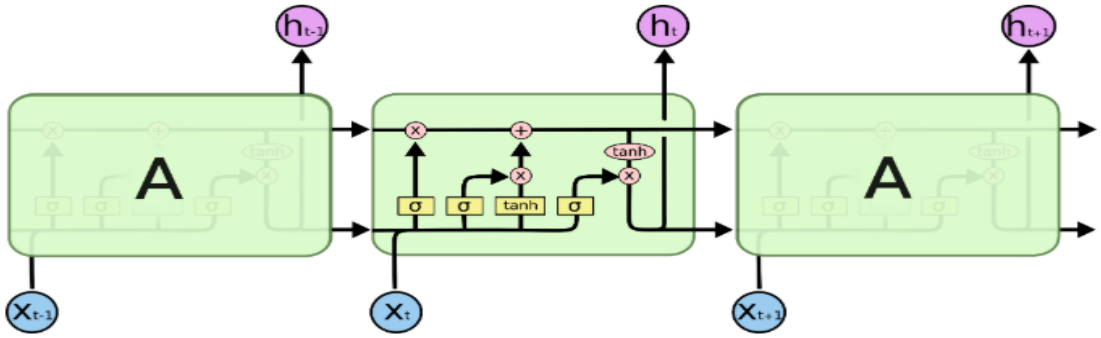


An unrolled recurrent neural network.

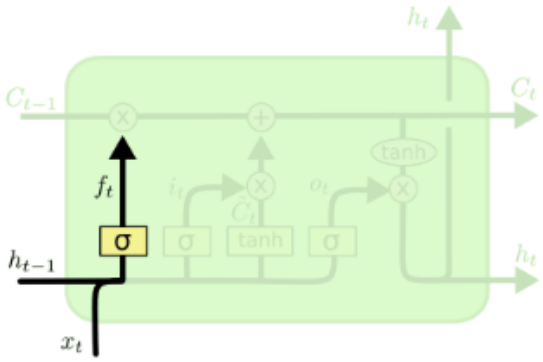
So the LSTM transmits data from module to module and, for example, for generating Ht we use not only Xt, but all previous input values X.

2.4 Working of LSTM models:

Discovered in 1997 by Hochreiter & Schmidhuber, LSTM's are a special kind of RNN capable of learning long term dependencies. Remembering information for long period of time is their default behavior.



One LSTM cell is consist of 4 layers interacting in a very special way. The key element of LSTM is the cell state, it is the horizontal line running through the top of the diagram. It runs straight down to entire chain with some minor linear interactions. The LSTM's also have the capacity to add or remove information with the help of regulated structures, normally known as gates. Gates are composed of a sigmoid linear layer followed by a pointwise multiplication. The output of sigmoid layer lies between 0 and 1, which describes how much information will flow across the chain. The 0 output will imply that no information will flow while 1 output tells that everything will pass through.

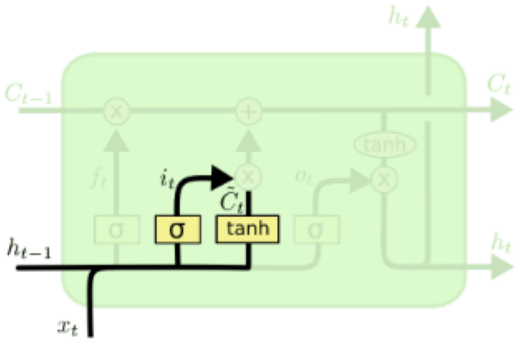


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

fig. forget gate

The first step in a LSTM cell is to decide how much information we are going to throw away from the cell state. This is done through a forget gate.

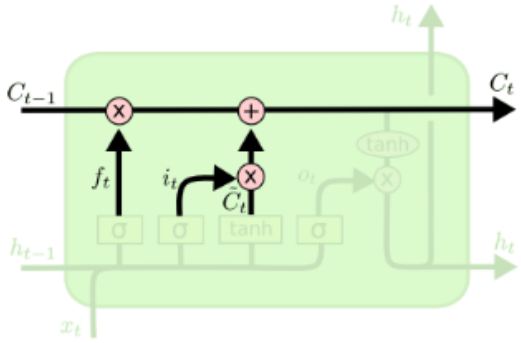
The next step in LSTM is to decide what new information we are going to store in the cell state. It has two parts, first is one sigmoid layer also known as input gate layer followed by a tanh layer which creates a vector of new candidate values.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

fig. input gate

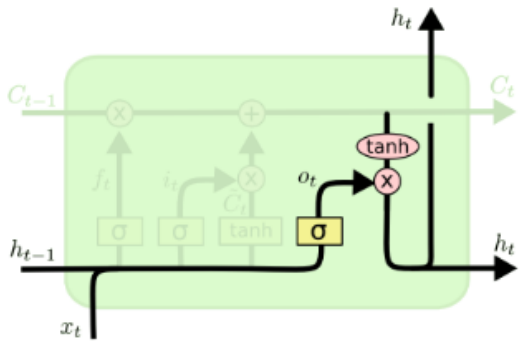
In order to update the cell state we multiply the old state by f(t) and then we add the the output of input layer(i(t) * c(t)) . This shows the new scaled candidate value.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

fig. scaled output

As a final step we need to decide the output, the output is based on the cell state but it is a filtered version. First we pass the input of the previous cell to a sigmoid layer. Output of the cell state is pass through a tanh layer which pushes the value between (-1 and 1) and multiply with the output of sigmoid gate so that we will only output the desired part.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

fig. output gate

2.5 Sequence to Sequence Model

Now we're ready to move to sequence to sequence models (also called seq2seq). The basic seq2seq model consists of two RNNs: an encoder network that processes the input and a decoder network that generates the output. Hence as the output of decoder net, we get the translated information in targeted language. However, let's think about one trick. Google Translate currently supports 103 languages, so we should have 103x102 different models for each pair of languages. Of course, the quality of these models varies according to the popularity of languages and the amount of documents needed for training this network. The best that we can do is to make one NN to take any language as input and translate into any language.

3. Model Architectures:

The implementation is based on the paper "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". They have proposed a common sequence-to-sequence learning framework with attention. It has three components: an encoder network, a decoder network, and an attention network. But in my implementation, for the purpose of simplicity, only encoder and decoder networks are used. The use of attention layer has been omitted. The encoder transforms a source sentence into a list of vectors, one vector per input symbol. Given this list of vectors, the decoder produces one symbol at a time, until the special end-of-sentence symbol (EOS) is produced. In the actual paper stacked layers of LSTM has been used and has been trained on multiple GPU.

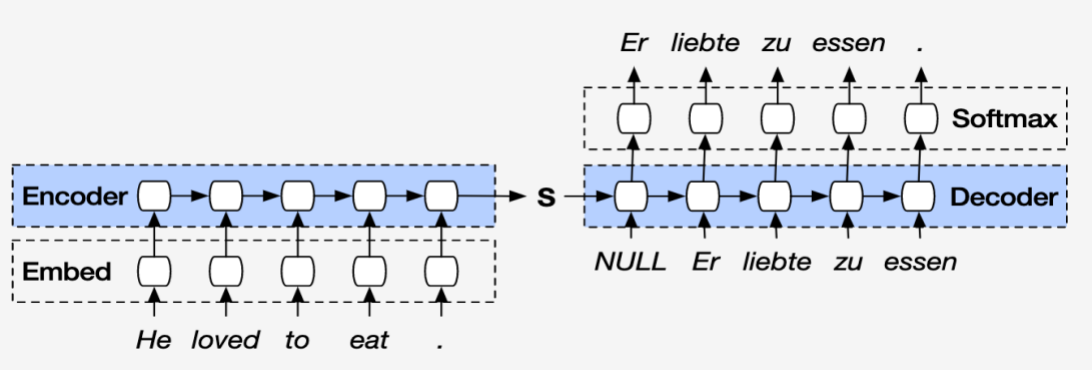


fig. Simplified architecture

In my architecture only one LSTM encoder and only one LSTM decoder has been used and a repeat vector is used in between to repeat the input vector while training. Word embeddings are used for converting a given sequence into a one hot encoded vector. This step is necessary as every model works only on numerical data so we need some way to convert it into a numerical data. And the output is a time distributed layer because learning algorithm used for LSTM are Backpropagation through time(BPTT).

A rough diagram for the architecture is as follows:

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 10, 256)	1466368
lstm_1 (LSTM)	(None, 256)	525312
repeat_vector_1 (RepeatVecto	(None, 5, 256)	0
lstm_2 (LSTM)	(None, 5, 256)	525312
time_distributed_1 (TimeDist	(None, 5, 3670)	943190

fig. Model Architecture

4. Results and future improvements:

The model has been trained for translating German to English, The dataset is available at http://www.manythings.org/anki/deu-eng.zip. It contains 152,820 pairs of English to German phases, one pair per line with a tab separating the language, which is supposed to be a fair amount of data to build a demo translation model. Initially the model was trained on 10000 phrases, later I trained on 50000 phrases and the model was able to produce the translated English phrase but it is not up to the mark. The model has been trained for 30 epochs with a batch size of 64, with one epoch taking almost 10 minutes to run.

The sample output is as follows:

```
testing the Model..
source=[ich bin reif], target=[im mature], predicted=[im am]
source=[frohe ostern], target=[happy easter], predicted=[no everyone wrong]
source=[verlieren sie nie die hoffnung], target=[never lose hope], predicted=[no smoking no]
source=[uns fehlt es an nichts], target=[we lack nothing], predicted=[were be]
source=[welche hohe hat es], target=[how high is it], predicted=[crows asian of]
source=[tom war groartig], target=[tom was terrific], predicted=[tom was loved]
source=[drucken sie tom], target=[hug tom], predicted=[never tom tom]
source=[tom hatte einen schlaganfall], target=[tom had a stroke], predicted=[tom had a deported]
source=[ich werde brav sein], target=[ill be nice], predicted=[ill be cats]
source=[lassen sie sich nicht tauschen], target=[dont be deceived], predicted=[dont you see me]
source=[tom hat das getan], target=[tom did this], predicted=[tom lost me]
source=[ich konnte nicht antworten], target=[i couldnt answer], predicted=[i night be]
source=[was macht deine erkaltung], target=[hows your cold], predicted=[what do we see]
source=[so was tun wir nicht], target=[we dont do that], predicted=[isnt do do do]
source=[ich werde allein sein], target=[ill be alone], predicted=[ill be it it]
source=[tom sieht mude aus], target=[tom looks sleepy], predicted=[tom looks a]
```

fig. Sample Results

It can be seen that model is able to predict few of the correct words while it's showing wrong translation for so many words. Specially when the length of the sentence is long, the model is failing to predict the actual translation. so there's still too much of room to improve.

As per my intuition from the paper, below are the improvements I am planning to work on:

- **Architecture:** Try different architecture for Neural machine Translation for example LSTM with attention mechanism which perform better for Translation tasks as explained in Google Neural Machine Translation paper.
- **Pre-trained weight vectors:** In my approach the vocabulary has been generated only from the dataset availabale, so there's high probability of so many missing words. In order to overcome this we can use a pre-trained word vector and it will definitely increase the size of vocabulary.
- **More Data**: Dataset used for training can be extended to 100,000 sentences or even more.
- **Layers** : The encoder and/or the decoder models could be expanded with additional layers and trained for more epochs, providing more representational capacity for the model.
- **Input Order**: The order of input phrases could be reversed, which has been reported to lift skill, or a Bidirectional input layer could be used.

- **Regularization** :The model could use regularization, such as weight or activation regularization, or the use of dropout on the LSTM layers.

- **Units**: The number of memory units in the encoder and decoder could be increased, providing more representational capacity for the model.

5. Applications

Machine translation is used extensively at many places as it serves the as the bridge between different languages. Some of the application are as follows:

a) Machine translation is used by Google for translating one language to another.
b) Machine translation can be used as a bridge between existing technologies in some language to some other language. Such as if Chatbot is made for English language interactions but it can now be extended to other languages too as simply convert other languages to English first then get a reply from the Chatbot and convert that reply again to the source language. Another example is IBM Watson, it is primarily made for English language only but now it can be extended to other languages too by the help of machine translation.
c) The general public uses language translator for inter-language communication.

References:

- https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html
- https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
- https://machinelearningmastery.com/develop-encoder-decoder-model-sequence-sequence-prediction-keras/
- Google's neural machine translation system: Bridging the gap between human and machine translation.