

UE17CS338

Topics in Deep Learning

Auto-Encoder with Attention for Neural Machine Translation

Project Report

Vinayaka M Hegde PES1201701600

S Thejas PES1201701621

Ramakrishnan K PES1201701906

Table of Contents

[**Project Abstract and Scope**](#_939f5mlsat8q) **3**

[**Preprocessing**](#_nf67koobgblx) **3**

[**Architecture**](#_cgdqdhfko2x0) **4**

[**Testing and Results**](#_vg88sd7lwq6l) **6**

[**Constraints, Assumptions and Dependencies**](#_zdlriuwsgztw) **8**

[**Future Work Plan**](#_err4m2ilm147) **8**

[**References**](#_7jc9l6vnjdxy) **8**

# Project Abstract and Scope

This project aims at building a robust Neural Machine Translation model. A custom model is built using tensorflow to translate English to French. The model aims at overcoming the problem of handling context through very long sentences encountered during translation using Bahdanau attention. Further improvements are made on the model using Drop-out technique to reduce overfitting. The model is then compared with a baseline model without attention and the results are plotted. The model is a seq2seq model built using RNN-LSTM as encoder and decoders.

The inspiration for this model comes from the set-back of the existing LSTM model: Vanilla LSTM model can not remember the context of a word if its occurrences are far apart. To overcome this problem attention is being used. Attention performs better even in case of smaller sentences as the context is embedded into the code generated by the encoder. Currently the model handles translation from english to french. Given more data, the model can be extended to other languages as well.

# Preprocessing

The input contains a list of sentences separated by \n character. These sentences are provided in two text files - one in English and other in French. So, first we iterate over sentences in a text file and create two lookup tables - First one is called vocab\_to\_int that has mapping from words to index and second one is called int\_to\_vocab that has mapping from index to words. These are used in the embedding phase. There are four special tokens added to these lookup tables:

<PAD> - Token used to pad the variable length sentences.

<EOS> - Indicates the end of the decoder output.

<UNK> - This is used when a word that does not belong to vocabulary is generated by decoder.

<GO> - Indicates the start of decoder output.

Finally, in the embedding phase, we convert every word to a vector of size embedding\_size that goes as input to the model.

# Architecture

In this project we implement a seq2seq model for translating sentences from English(input) to French(output) using Recurrent Neural Network(RNNs) with Long Short Term Memory(LSTM) cells.

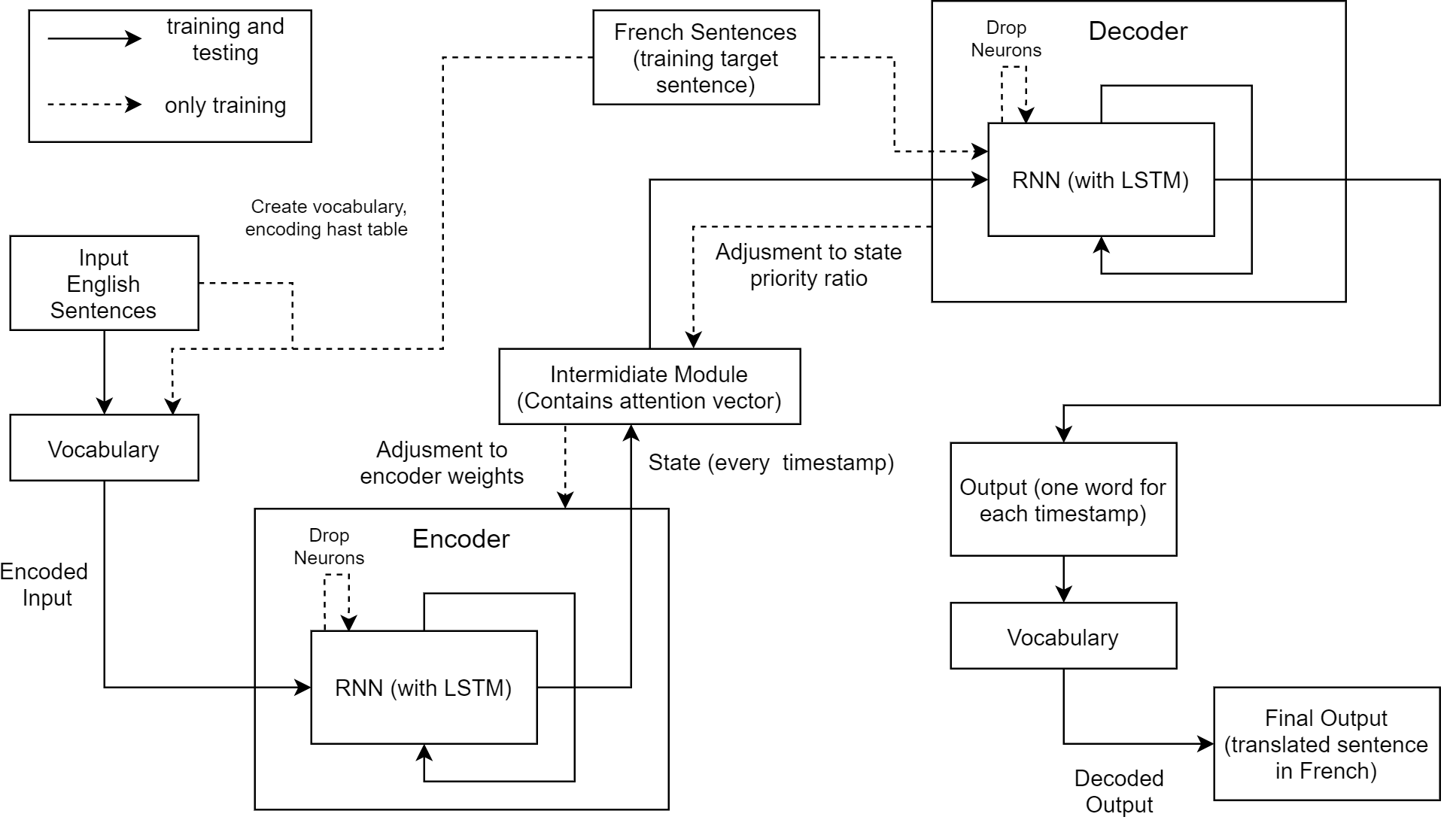
Seq2seq is a model that takes a sequence as input and produces another sequence as output. It consists of an encoder component, which efficiently compresses and encodes the input sequence into a code, and a decoder component which takes the code as the input and produces the translated sequence. During training the encoder-decoder component tries to ensure that the encoded information can be decoded correctly.

RNN is a class of Artificial Neural Networks(ANN), where the connections between the neurons of the network form a directed graph along a temporal(time-varying) sequence. In our model, we feed the current input from a sequence along with a cell state which captures the encoding of the previous input in the sequence. There are many types of RNN. Due to their performance and handling of the Vanishing Gradient problem, the LSTM and the GRU variant are quite familiar. The prime difference between them is that GRU has a lower number of gates, making it simpler. The reason LSTM is being used over GRU in our project is that LSTM works better with longer sentences when compared with GRU.

LSTMs have the ability to remember the most recently encountered information. This is done using the cell state which goes through the entire network. However, LSTM cannot work with very long sentences. This drawback is overcome by using Attention.

Attention is a mechanism of associating a word along with its context. The Bahdanau attention is used for this purpose to pay attention to only those words that affect the next output. Before, the previous output of the decoder and the hidden state was used to compute the current output of the decoder. But after using attention, we use a context vector (that has the combined information from all the encoder states) also while computing the current output. We have also used Teacher Forcing Technique. It is a method where we feed the expected (target) word to the next decoder time instance so that the subsequent outputs that are generated will be more accurate to the context. It basically computes the probability of next output, given the previous hidden state and previous target outputs.

The architecture of the model



The model takes the English sentences as input. The input is then passed word by word into the encoder RNN. The encoder RNN maintains the state at every timestamp and passess it on to the intermediate module. And when encoder RNN is done, it generates a <SOS> token to indicate the beginning of the output.

In the intermediate module, we first implement the attention mechanism from Tensorflow’s seq2seq.BahdanauAttention. Then we use seq2seq.AttentionWrapper to wrap this attention mechanism to the decoder.

The code which now has the attention vector embedded in it, is then passed on to the decoder layer along with the hidden states of the encoder. When the decoder is done generating output, it emits a <EOS> token to mark the end of the output. During training the decoder compares the required output with the predicted output to update the weights. In order to prevent overfitting, drop-out technique is used. In this technique, during training, for each of the inputs, few of the neurons are drop-out of the training process. The number of neurons dropped is a hyper parameter called keep probability. This makes the model more robust. This phase of weight update is optimised using RMSPropOptimiser.

In the encoder and decoder LSTMs, the hidden state and cell states are computed using:

Ht = Ot \* tanh(Ct)

Ct = Ct-1 \* ft + it \* gt

where,

Ht = hidden state

Ct = cell state

it , gt = used for write gates

ft = forget gate

For Bahdanau attention mechanism, attention weights are computed using :

res = tanh ( Wencoder \* Hencoder + Wdecoder \* Hdecoder)

alignment scores = Walign \* res

attention weights = softmax (alignment scores)

where,

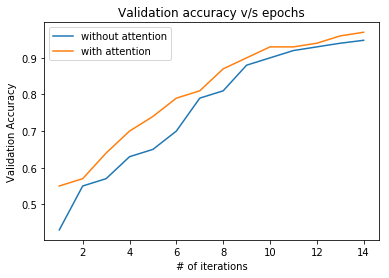
Wencoder , Wdecoder = weights of encoder and decoder

Hencoder, Hdecoder = hidden states of encoder and decoder

Walign = weight vector for alignment scores

# Testing and Results

The model developed displayed a good increase in accuracy. The best keep probability was found to be 0.5. The same model without attention was able to bring out a 94% accuracy. Adding attention improved the accuracy to 97%.

The effect of the hyper-parameters on the performance of the model is summarised in the following plots:

It is clear from these plots that the model performs better with attention. This is because attention helps capture the context of the word better, thereby improving the performance.

# 

For drop-outs, 50% probability gives highest accuracy. A higher value could cause overfitting when used in hidden layers. A lower value could prevent the neurons from learning, since many would be dropped out.

# 

# Constraints, Assumptions and Dependencies

Constraints :

1. Our model encounters difficulties when faced with rare words or words that are ambiguous as sometimes the model cannot interpret information to the required context.

Assumptions :

1. Only ASCII characters are given as input.
2. The english sentences and the translated french sentences only contain words which were present in the training set.

Dependencies :

1. Tensorflow module version 1.14.0

# Future Work Plan

1. Test the functionality of the network with other languages.
2. Test the functionality of this architecture for summarisation of text.

# References

1. FloydHub Blog. 2020. *Attention Mechanism*. [online] Available at: <<https://blog.floydhub.com/attention-mechanism/>> [Accessed 21 April 2020].
2. TensorFlow. 2020. *Introduction To Tensorflow*. [online] Available at: <<https://www.tensorflow.org/learn>> [Accessed 21 April 2020].
3. Pathmind. 2020. *A Beginner's Guide To Attention Mechanisms And Memory Networks*. [online] Available at: <<https://pathmind.com/wiki/attention-mechanism-memory-network>> [Accessed 21 April 2020].
4. Arxiv.org. 2020. [online] Available at: <<https://arxiv.org/pdf/1409.0473.pdf>> [Accessed 21 April 2020].
5. Medium. 2020. *Understanding Encoder-Decoder Sequence To Sequence Model*. [online] Available at: <<https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346>> [Accessed 21 April 2020].