Sequence to Sequence Model: Video Caption Generation

*using TensorFlow in python

By- Abhinav

Abstract*— This work is to learn and understand how the sequence to sequence (s2s) neural networks learning. The implementation will give a clear picture of how the s2s helps in speech recognition, video description, language translation, image captioning etc. Keywords—*sequence to sequence, LSTM, Luong Attention Mechanism etc.

I. Introduction

S2S are special problems where inputs and outputs are sequences. So it is like using input sequences and converting them into output sequences. It can be used in language translation where input could be some language like *Hindi* and output will be some other language like *English*. This translation is done sequence by sequence and so it is called sequence to sequence problem.

In this work, we have used this s2s model to determine the video captions. So given the feature vector for every frame of the video, s2s training of the model is required to determine the captions for each of the videos. The input and output sequences are of different length so we require an encoder and a decoder to determine the captions completely.

II. Problem Description

Dataset:

We have video description in the form of features which is a float value of size (80,4096), which is 80 frames and 4096 features. There are 1450 videos and each of these videos have their own feature data. Every video has list of captions, on an average there are 16 captions each for every video.

Approach:

1) Preprocessing: The first step is to preprocess our data to make it usable during the training of the model. We already have feature vector which is a fixed length informative vector for every video and so we didn't have to do anything with these feature vectors. We need to transform our captions into some sort of learning vectors so that our network can understand it and could use it further in determining the captions for each video.

There are different ways of encoding the captions. We can do One Hot Encoding, where categorical data are converted into a form which can be further used by the network. It is in the form of 0 and 1. The problem with this is that it doesn't store the sequence of the words and for a caption which is 40 words long it will have 40x40 dimension of encoding. The second approach could be to use Integer mapping where each words are represented by unique integer; this is a better approach but the problem with this approach is that it cannot learn the sequence of the words in the captions. I have used Embedding mapping, this is because it reduces the dimensionality of the captions and meaningfully represent captions in the transformed space.

So the captions for every videos have different and similar words, so we need to store the unique word from every caption and store it as a vocabulary which can be used further in determining the embedding of the captions.

2) LSTM Model: Recurrent Neural Network (RNN) works well when it has to predict the next possible word or output based on the previous one, but when it comes to predict something that might depend on the output which was further back. It is quite possible that the gap between the two words, the word needed to be predicted and the word based on which this prediction will be done, becomes very large. And as that gap grows it becomes difficult for RNN to predict that relevant information.

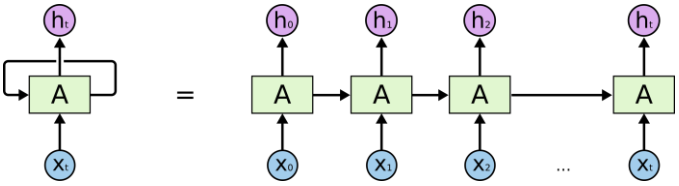

Fig1. RNN model

LSTM are special kind of RNN, that overcomes this limitation of RNN. LSTM or Long Short Term Memory, remembers information for long periods of time.

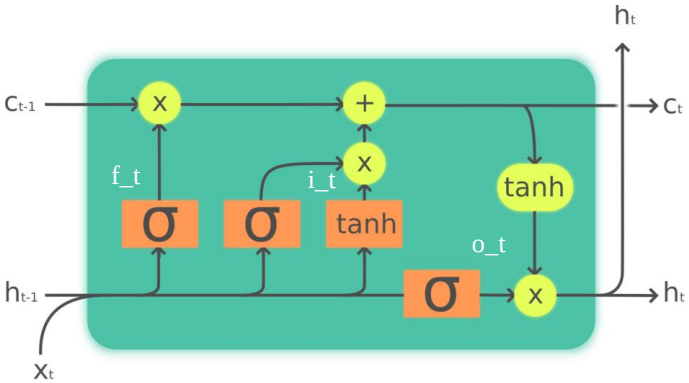

Fig2. There are four neural network in LSTM

LSTM is made up of three gates:

- Forget Gate (f_t): Controls if/when the context is forgotten
- Input Gate (i_t): Controls if/when a value should be remembered by the context
- Output Gate (o_t): Controls if/when the remembered value is allowed to pass from the unit

LSTM model is further classified into Encoder and Decoder LSTM. The logic behind this is the variation in the length of the inputs and outputs. *Encoder LSTM:* It reads the input sequence and encoding it into a fixed length. *Decoder LSTM:* It decode the fixed length encoded vector and output the predicted sequence.

3) Attention Mechanism: It helps by mapping all the hidden state of the input sequences and decode it. It creates mapping between each time step of the decoder output to all the encoder hidden state. So, for every output from the decoder it maps the input sequences that were used and select the specific sequence to generate the final output.
There are two types of Attention Mechanism, *Bahdanau Attention and Luong Attention.* For this project I have used Luong Attention.

Luong Attention Process:

1. Encoder Hidden State: Encoder produces a hidden state for each element in the input sequence
2. Decoder RNN: Hidden state produced in the last time step and word embedding from the last step produces a new hidden state which will be used in the subsequent steps
3. Scoring function: Calculates the alignment score using the encoder outputs and decoder hidden state
4. Softmax the Alignment score: Softmax the weight so that it is between 0 and 1
5. Calculating the Context Vector: Attention weights are multiplied with the encoder outputs.
6. Producing the Final Output: Obtains the probability of the next predicted word.

4) Loss: Cross entropy to measure the probability error between logits and labels.

5) Optimizer: Adam Optimizer is used to for adaptive learning and improving the accuracy

III. Result

The training of the model requires some correction in the code because of which I have not got the actual result. However, I have understood the whole process of sequence to sequence neural network and how it can help us further in different problems like video description, language translation etc.
I will update the code and document once my code starts to show some results.

IV. Conclusion

We used sequence to sequence modelling using LSTM cells and also used attention mechanism to focus on the input sequences so that we get more appropriate output.

References

[1] https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

[2] https://web.stanford.edu/class/cs379c/archive/2018/class_messages_listing/content/Artificial_Neural_Network_Technology_Tutorials/OlahLSTM-NEURAL-NETWORK-TUTORIAL-15.pdf

[3] https://blog.floydhub.com/attention-mechanism/

[4] https://www.youtube.com/watch?v=wY0dyFgNCgY

[5] https://github.com/llSourcell/seq2seq_model_live/blob/master/2-seq2seq-advanced.ipynb

[6] https://github.com/tensorflow/nmt