Introduction to Sequence Models — RNN, Bidirectional RNN, LSTM, GRU

towardsdatascience.com/introduction-to-sequence-models-rnn-bidirectional-rnn-lstm-gru-73927ec9df15

Rohith Gandhi June 26, 2018

A brief explanation

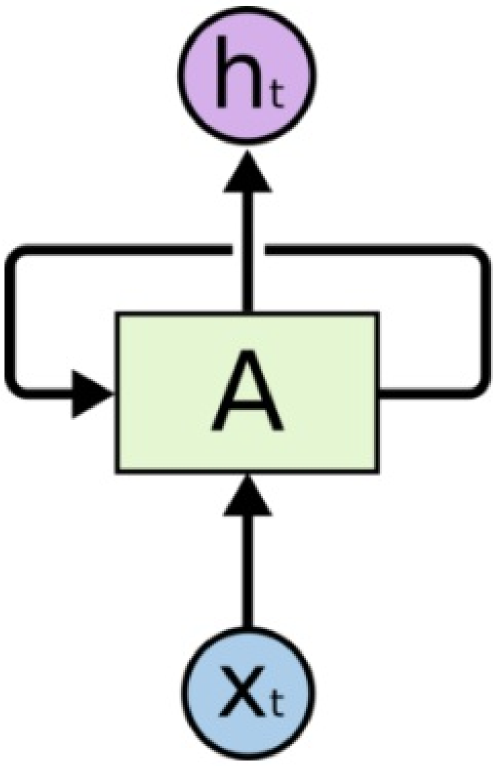
Introduction — Why do we need Sequence Models??

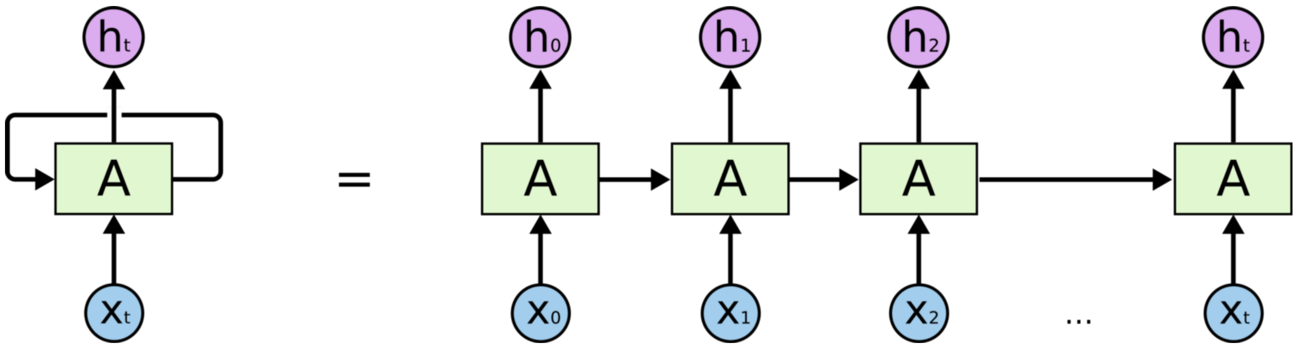
Why do we need sequence models when we already have feedforward networks and CNN? The problem with these models is that they perform poorly when given a sequence of data. An example of sequence data is an audio clip which contains a sequence of spoken words. Another example would be a sentence in English which contains a sequence of words. Feedforward networks and CNN take a fixed length as input, but, when you look at sentences, not all are of the same length. You could overcome this issue by padding all the inputs to a fixed size. However, they would still perform worse than an RNN because those conventional models do not understand the context of the given input. This is where the major difference between sequence models and feedforward models lies. Given a sentence, when looking at a word, sequence models try to derive relations from the previous words in the same sentence. This is similar to how humans think as well. When we are reading a sentence, we don't start from scratch every time we encounter a new word. We process each word based on the understanding of the previous words we have read.

Recurrent Neural Network

1. Recurrent Neural Network

The recurrent neural network is represented as shown in the above figure. Each node at a time step takes an input from the previous node and this can be represented using a feedback loop. We can unfurl this feedback loop and represent it as shown in the figure below. At each time step, we take an input x_i and a_i-1(output of the previous node) and perform computation on it and produce an output h_i. This output is taken and given to the next node. This process continues until all the time steps are evaluated.





2. Recurrent Neural Network

The equations describing how the outputs are calculated at each time step is represented below.

$$\text{Let } a_t \text{ represent the ouput from the previous node}$$

$$a_t = f(h_{t-1}, x_t)$$

$$g(x) = \tanh x$$

$$a_t = g(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t)$$

$$a_t = \tanh W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t$$

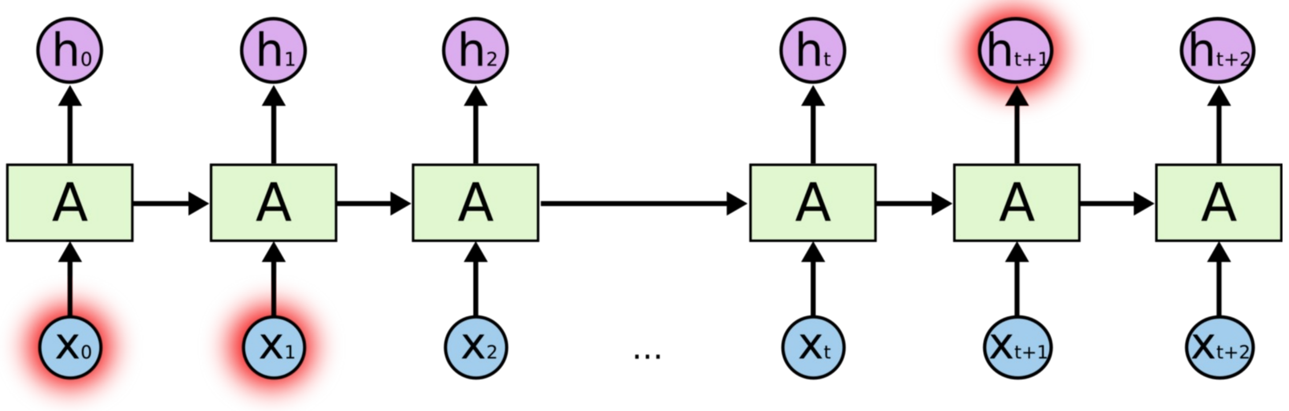
$$h_t = W_{hy} \cdot a_t$$

3. Recurrent Neural Network

Backpropagation in recurrent neural networks occurs in the opposite direction of the arrows drawn in figure 2. Like all other backpropagation techniques, we evaluate a loss function and obtain gradients to update our weight parameters. The interesting part of backpropagation in RNN is that backpropagation occurs from right to left. Since the parameters are updated from final time steps to initial time steps, this is termed as backpropagation through time.

Long Short-Term Memory — LSTM Network

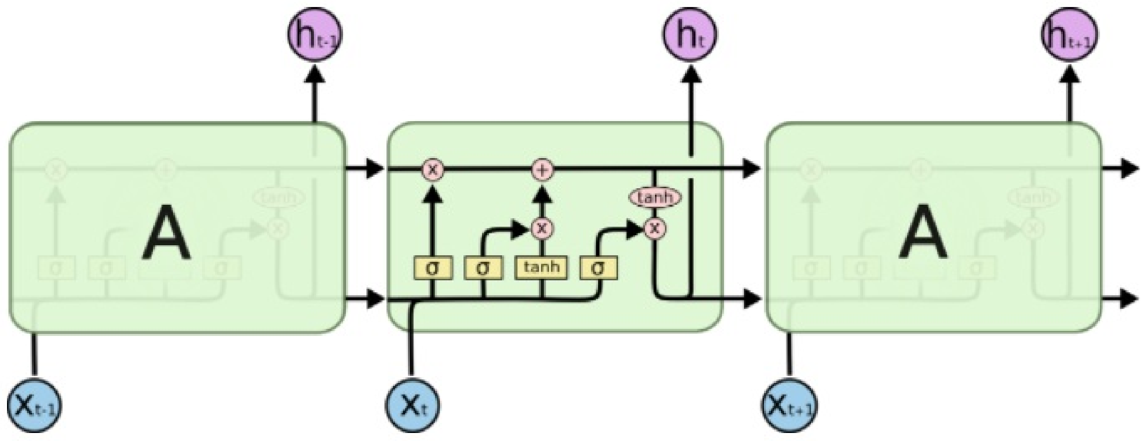
The disadvantage with RNN is that as the time steps increase, it fails to derive context from time steps which are much far behind.



4. Recurrent Neural Network

To understand the context at time step t+1, we might need to know the representations from time steps 0 and 1. But, since they are so far behind, their learned representations cannot travel far ahead to influence at time step t+1. Ex: "I grew up in France ….. I speak fluent French", to understand that you speak French, the network has to look far behind. But, it is not able to do so and this problem can be attributed to the cause of vanishing gradients. Therefore, RNN is able to remember only short-term memory sequences.

To solve this problem, a new kind of network was introduced by Hochreiter & Schmidhuber called Long Short-Term Memory.

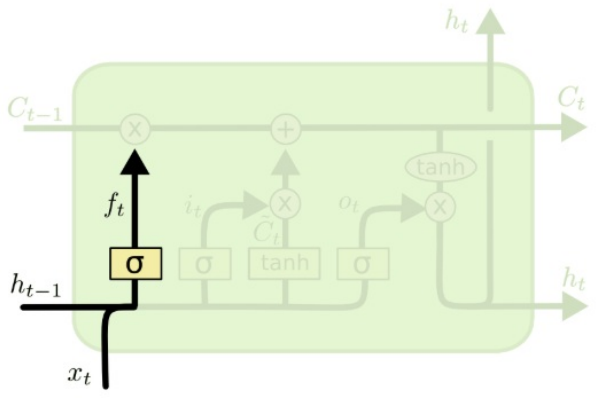


The repeating module in an LSTM contains four interacting layers.

5. LSTM

The structure of an LSTM network remains the same as an RNN, whereas the repeating module does more operations. Enhancing the repeating module enables the LSTM network to remember long-term dependencies. Let's try to break down each operation which helps the network remember better.

1. Forget gate operation

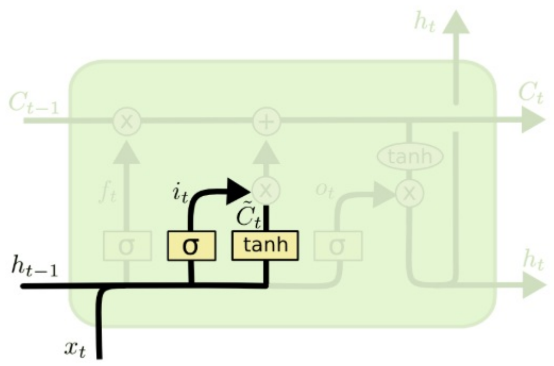


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

5. Forget operation

We take the input from current time step and the learned representation from previous time step and concatenate them. We pass the concatenated value into a sigmoid function which outputs a value(f_t) between 0 and 1. We do an element-wise multiplication between f_t and c_t-1. If a value is 0, then it is eliminated from c_t-1, if the value is 1, then it is completely let through. Therefore, this operation is also called “Forget gate operation”.

2. Update gate operation

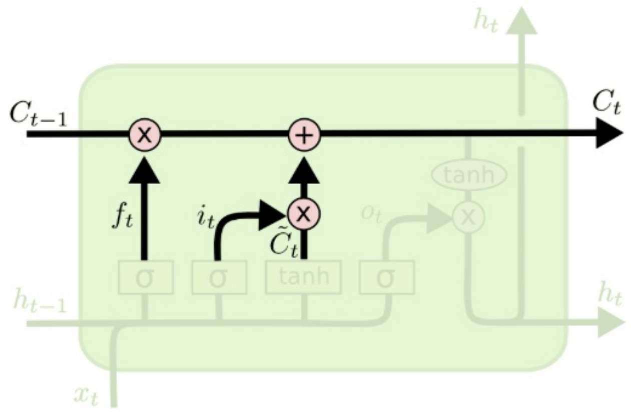


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

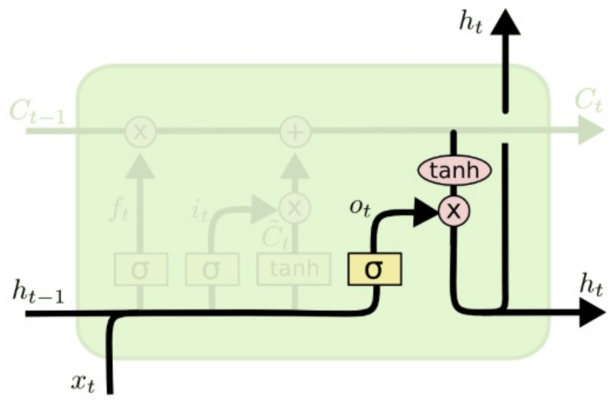
6. Update operation

The above figure represents the “Update gate operation”. We concatenate values from current time step and the learned representation from previous time step. By passing the concatenated values through a tanh function we generate candidate values and by passing it through a sigmoid function we choose which values to be selected from the candidates. The chosen candidate values are updated to c_t-1.

3. Output gate operation



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



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

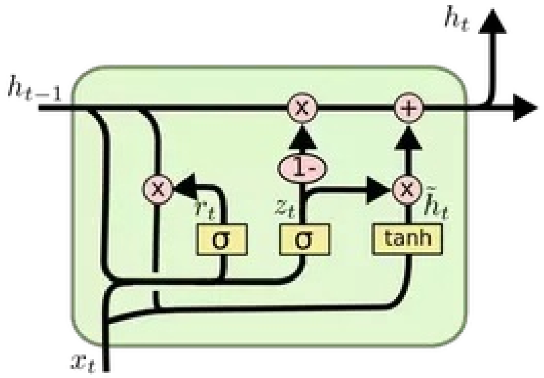
7. Updating the values(left) and Output operation(right)

We concatenate values from current time step and the learned representation from previous time step and pass it through a sigmoid function to choose which values we are going to use as the output. We take the cell state and apply a tanh function and do an element-wise operation which lets through only the selected outputs.

Now, this is a lot of operations to be done in a single cell. When using a bigger network, the training time would significantly increase compared to an RNN. There is an alternative to LSTM if want to reduce your training time but also use a network that remembers long term dependencies. It is called Gated Recurrent Unit(GRU).

Gated Recurrent Unit — GRU Network

A GRU unlike an LSTM network does not have a cell state and has 2 gates instead of 3(forget, update, output).



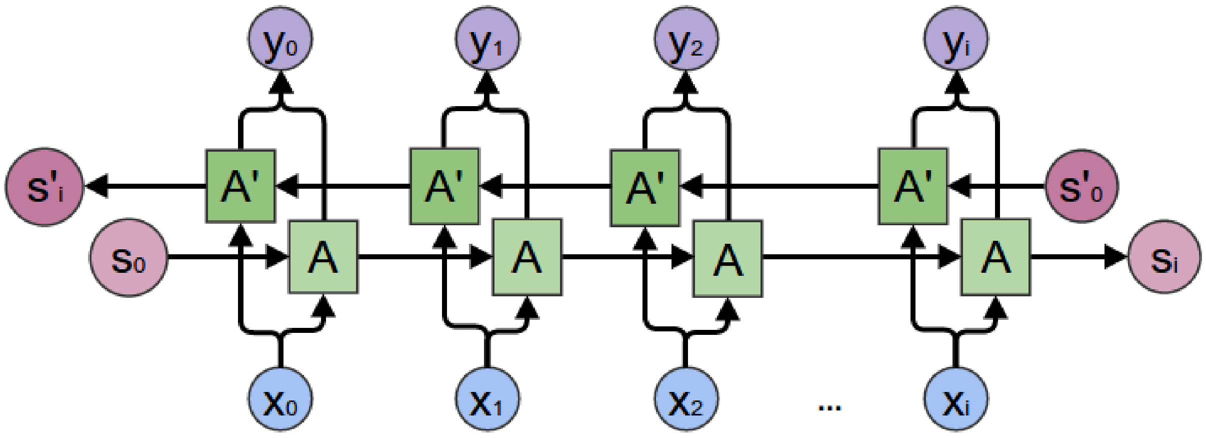
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

8. Gated recurrent Unit

A gated recurrent unit uses an update gate and a reset gate. The update gate decides on how much of information from the past should be let through and the reset gate decides on how much of information from the past should be discarded. In the above figure z_t represents the update gate operation, whereby using a sigmoid function, we decide on what values to let through from the past. h_t represents the reset gate operation, where we multiply the concatenated values from the previous time step and current time step with r_t. This produces the values that we would like to discard from the previous time steps.

Even though GRU is computationally efficient than an LSTM network, due to the reduction of gates, it still comes second to LSTM network in terms of performance. Therefore, GRU can be used when we need to train faster and don't have much computation power at hand.

Bidirectional RNN



9. Bidirectional RNN

A major issue with all of the above networks is that they learn representations from previous time steps. Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity. Take the following examples, "He said, Teddy bears are on sale" and "He said, Teddy Roosevelt was a great President". In the above two sentences, when we are looking at the word "Teddy" and the previous two words "He said", we might not be able to understand if the sentence refers to the President or Teddy bears. Therefore, to resolve this ambiguity, we need to look ahead. This is what Bidirectional RNNs accomplish.

The repeating module in a Bidirectional RNN could be a conventional RNN, LSTM or GRU. The structure and the connections of a bidirectional RNN are represented in figure 9. There are two type of connections, one going forward in time, which helps us learn from previous representations and another going backwards in time, which helps us learn from future representations.

Forward propagation is done in two steps:

- We move from left to right, starting with the initial time step we compute the values until we reach the final time step
- We move from right to left, starting with the final time step we compute the values until we reach the initial time step

Conclusion

Combining Bidirectional RNN with LSTM modules can significantly improve your performance and when you conflate them with an attention mechanism, you get state of the art performance in use cases such as machine translation, sentiment analysis etc. I hope this article was useful. There were a lot of mathematical equations involved and I hope it was not too intimidating. Let me know if have any doubts, I will try my best to sort it out :)

References

Understanding LSTM Networks -- colah's blog
*These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that…*colah.github.io