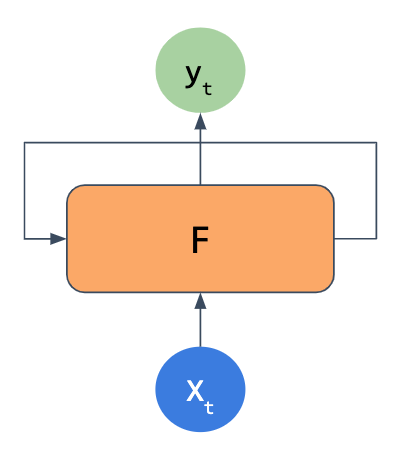
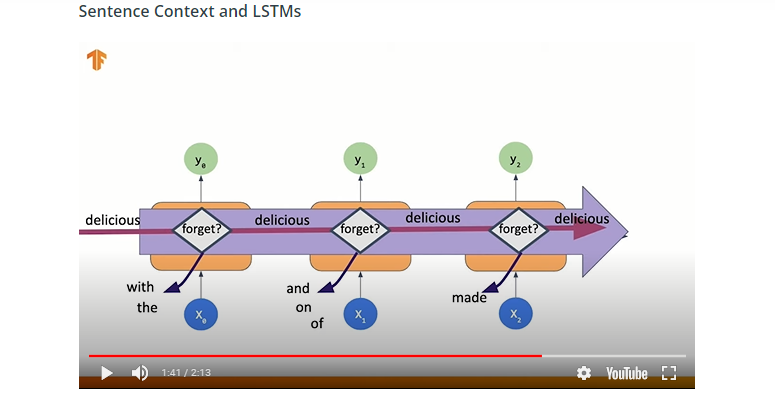
RNNs FOR NLP

Recurrent Neural Networks (RNNs) still take in some input x and output some y, but they also feed some of the output of the network back into itself. This may be done over and over, so that with text input, the network has some memory of words that came much earlier in a sequence.



The basic RNN flow

Sentence Context and LSTMs



he code for an LSTM layer itself is just the LSTM layer from tf.keras.layers, with the number of LSTM cells to use. However, this is typically wrapped within a Bidirectional layer to make use of passing information both forward and backward in the network, as we noted on the previous page.

*# A bidirectional LSTM layer with 64 nodes*

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64))

One thing to note when using a Bidirectional layer is when you look at the model summary, if you put in 64 LSTM nodes, you will actually see a layer shape with 128 nodes (64x2).

**No Need to Flatten**

Unlike our more vanilla neural networks in the last lesson, you no longer need to use Flatten or GlobalAveragePooling1D after the LSTM layer - the LSTM can take the output of an Embedding layer and directly hook up to a fully-connected Dense layer with its own output.

**Doubling Up**

You can also feed an LSTM layer into another LSTM layer. To do so, on top of just stacking them in order when you create the model, you also need to set return\_sequences to True for the earlier LSTM layer - otherwise, as noted above, the output will be ready for fully-connected layers and not be in the sequence format the LSTM layer expects.

*# Two bidirectional LSTM layers with 64 nodes each*

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64), return\_sequences=**True**)

tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64))

