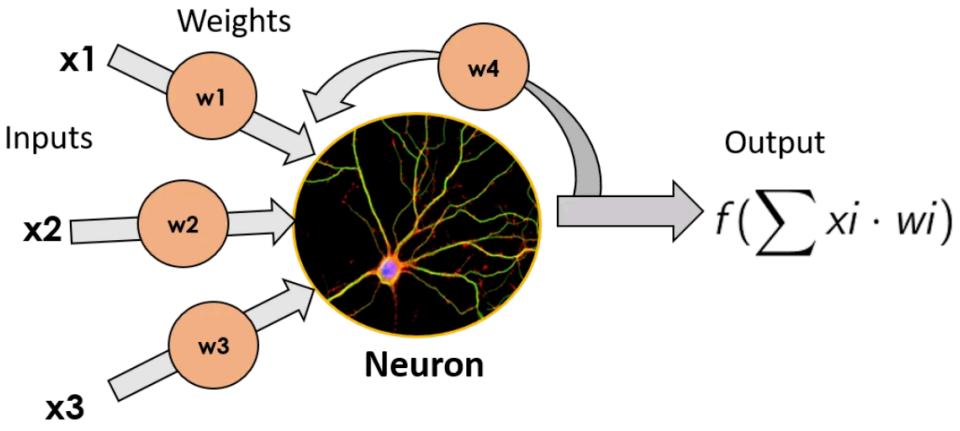
Beginner's Guide to RNN & LSTMs

Michael Wollowski

Summary of:

https://medium.com/@humble_bee/rnn-recurrent-neural-networks-lstm-842ba7205bbf

A Recurrent Neuron



Definition and Use of RNN

- Recurrent Neural Network is a feed-forward network that has an internal memory.
- They are designed to effectively deal with sequential data, such as text.
- For making a decision, an RNN considers the current input and the output that it has learned from the previous input.
- The internal state acts as (limited) memory.

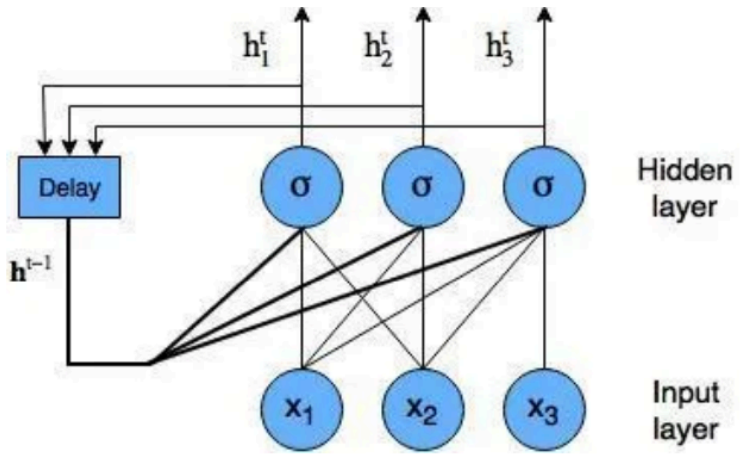
Difference Between FFNets and RNNs

- If using a a feed-forward neural network to process sequences:
- The entire sequence has to be presented.
- This is problematic because number of words in a sentence vary.
- Additionally, it treats the input more like a bag of words.
- It does not necessarily “perceive” the structure of the sentence.
- Think ”Mary loves John,” vs. “John loves Mary.”

Difference Between FFNets and RNNs

- People read sentences word by word.
- They keep prior words and the context in memory.
- They update their understanding based on the new words encountered.
- This is the basic idea of RNNs.
- They iterate through the elements of input sequence while maintaining a internal “state”.
- The internal state encodes everything it has seen so far.
- Admittedly, it is a fairly small state.

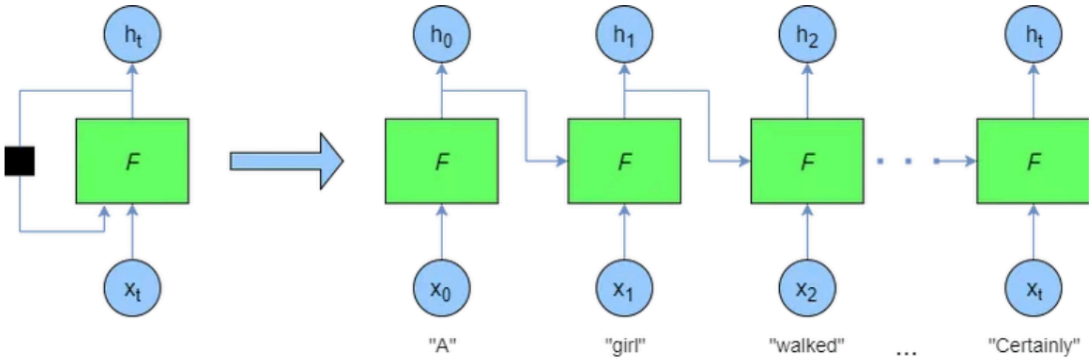
Excerpt of Architecture of an RNN



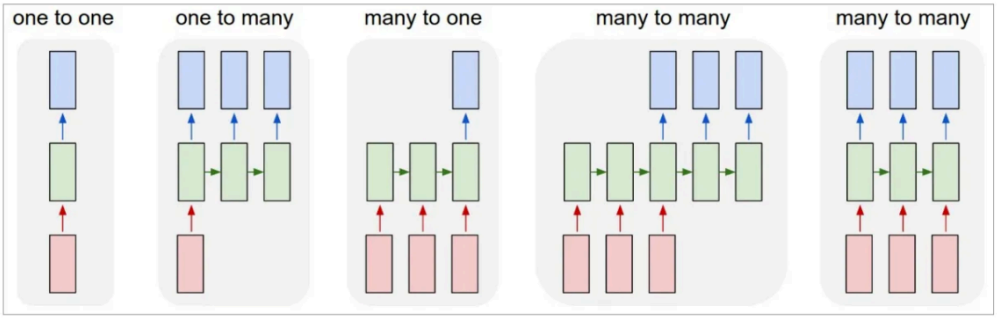
Example of RNN Processing

- Predicting next word.
- Consider the following text string: “A girl walked into a bar, and she said ‘Can I have a drink please?’
- The bartender said ‘Certainly { }”
- There are many options for the next word, as indicated by the curly brackets.
- It could be: “miss” or “ma’am.”
- However, other words could also fit, such as: “sir” or “mister.”
- In order to get the correct gender of the noun, the neural network needs to “recall” that two prior words designating the likely gender (i.e. “girl” and “she”) were used.

Unrolled RNN



RNNs for NLP



Each rectangle is a vector and arrows represent functions (e.g. matrix multiply). Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon). From left to right: **(1)** Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification). **(2)** Sequence output (e.g. image captioning takes an image and outputs a sentence of words). **(3)** Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment). **(4)** Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French). **(5)** Synced sequence input and output (e.g. video classification where we wish to label each frame of the video). Notice that in every case are no pre-specified constraints on the lengths sequences because the recurrent transformation (green) is fixed and can be applied as many times as we like.