**Recurrent Neural Network**

RNN are used for text classification. Here, current input sequences depends on previous sequences.

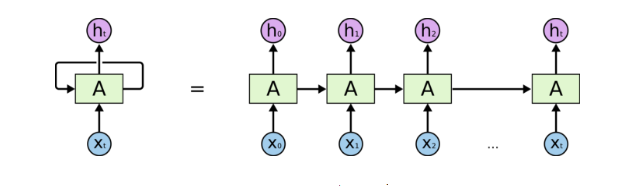


Figure 1 Multi-Layer RNN

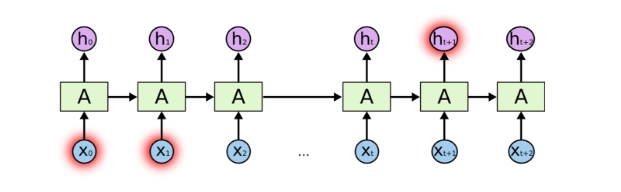


Figure 2 Problems Arising Due To Long Term Input Dependency in Basic RNN

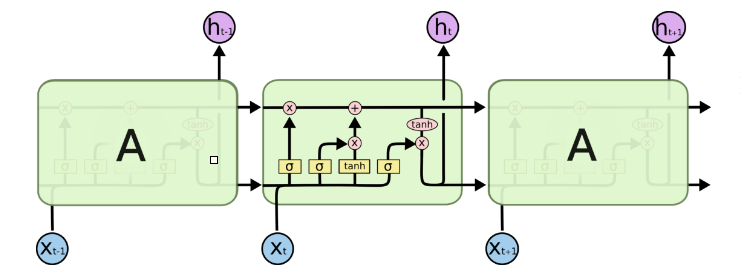


Figure 3 RNN Using LSTM

ht

ct-1

ct

Figure 4 Inside LSTM Network

tanh

it

ft

ot

Ct

ht-1

ht

s

tanh

s

s

xt

**Overview of LSTM Layers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Layer | Purpose | | Implementation | Reason for Working |
| Word Embedding | Represent words in form of vector |  | | Each sentence is represented in form of 1 hot encoded vectors containing weight matrix corresponding to it |
| Forget Gate | Forget data stored in previous state | ft = s (Wf. [ht-1,xt] + bf ) | | When we see a new subject, we want to forget the old subject |
| Input Gate | Decide what new information we’re going to store in the cell state | it = s(Wi .[ht-1,xt] + bi )  Ct = tanh ( Wc.[ht-1,xt] + bc ) | | *Sigmoid* layer decides which values we’ll update  *tanh* layer creates a vector of new candidate values, Ct, with which cell state can be updated |
| Cell State | Update the old cell state, Ct−1, into the new cell state Ct | Ct = ft\*Ct-1+it\*Ct | | *Sigmoid* layer which decides what parts of the cell state we’re going to output |
| Output Gate | Decide what will be the next hidden state | ot = s(Wo.[ht-1,xt] + bo)  ht = ot\* tanh(Ct) | | Put the cell state through *tanh* (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decide to. |