**RNN**

**History**

The **sequence of words** defines their meaning.

If we are trying to use the data where the sequence or order of their information matters, then we choose RNN.

RNN has **access to some prior input** to completely understand the sequence / sentence.

Objective is to identify the relationship between **successive inputs**.

Feed the **input to the hidden layers**.

Since each hidden layer has its own weights and activations, they behave independently (MLP)

To combine these hidden layers together, we shall have the same weights and bias for these hidden layers.

Combines hidden layers together, **weights and bias** are same.

At all the time steps weights of the recurrent neuron would be the same since it’s a single neuron now.

**How it works?**

**Inputs** are fed to the model

**RNN block**, applies something called as a recurrence formula to the **input vector** and its **previous state.**

**Current state =** 

Once all the **time steps**, calculate the **output (SoftMax to calculate probability)**.

**Errors are Backpropagated through time.**

**Unroll the network** at all the time steps.

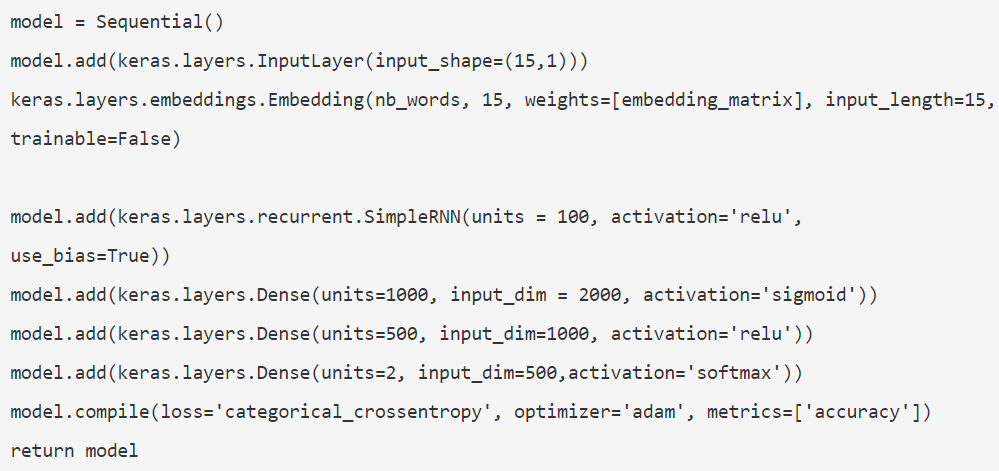
In an RNN we **may or may not have outputs** at each **time step.**

**Full sequence (Sentence)** as **one training example**, so the **total error (**cross entropy error) is just the sum of the **errors at each time step (Word).**

The weights are **then updated** for **both recurrent neuron** and the **dense layers after training One sentence.**

**Vanishing and Exploding Gradient Problem**

* Regular RNNs might have a difficulty in learning **long range dependencies**.
* Here we apply the chain rule and if any one of the gradients approached 0, all the **gradients would rush to zero** exponentially fast due to the multiplication.
* Such states would no longer help the **network to learn anything**.
* This is known as the vanishing gradient problem.



Units = no of hidden layers

LSTM

LSTM have the property of **selectively** **remembering patterns** for **long durations of time**.

LSTMs can **selectively remember or forget things.**

**LSTM works on Internal mechanisms** called **gates** that can **regulate the flow of information**.

These gates can learn which **data in a sequence** is **important** to **keep** or **throw** **away**.

By doing that, it can **pass relevant information** down the **long chain of sequences** to **make predictions**.

When you read the review, your **brain** **subconsciously** only **remembers** important **keywords**.

It can learn to **keep only relevant information** to make predictions, and **forget non relevant data.**

The **core concept** of LSTM’s are the **cell state**, and it’s **various gates.**

**Cell State** – transfers relative information – memory of the network – earlier time steps can also pass in the later time steps.

Information gets added or remove from the cell state during training through gates.

Gates are different networks – learn or decide which to keep or remove.

Gates – contains Sigmoid function, tanh

**A forget gate, input gate, and output gate.**

**Forget gate** (decides which information needs to be thrown away)

Input – previous hidden state + current input (sigmoid)

**Input gate** (decides which information needs to be update to the cell state)

Input – previous hidden state + current input (both sigmoid and tanh)

**Cell State**

**Point-wise multiplication** from forget gate

**Point-wise addition** from input gate

**Output Gate (decides what the next hidden state would be)**

**Input-** previous hidden state + current input (sigmoid)

**Input**-tanh (updated cell state)

The **new** **cell state** and the **new hidden** **state** is then carried over to the next time step.

**To review, the Forget gate decides what is relevant to keep from prior steps.**

**The input gate decides what information is relevant to add from the current step.**

**The output gate determines what the next hidden state should be.**

<https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>

<https://analyticsindiamag.com/how-to-code-your-first-lstm-network-in-keras/>

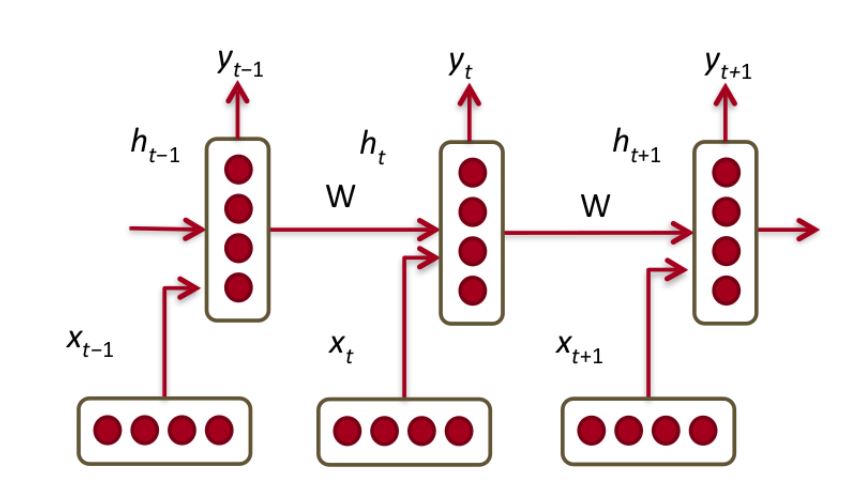
**Units in LSTM / RNN**

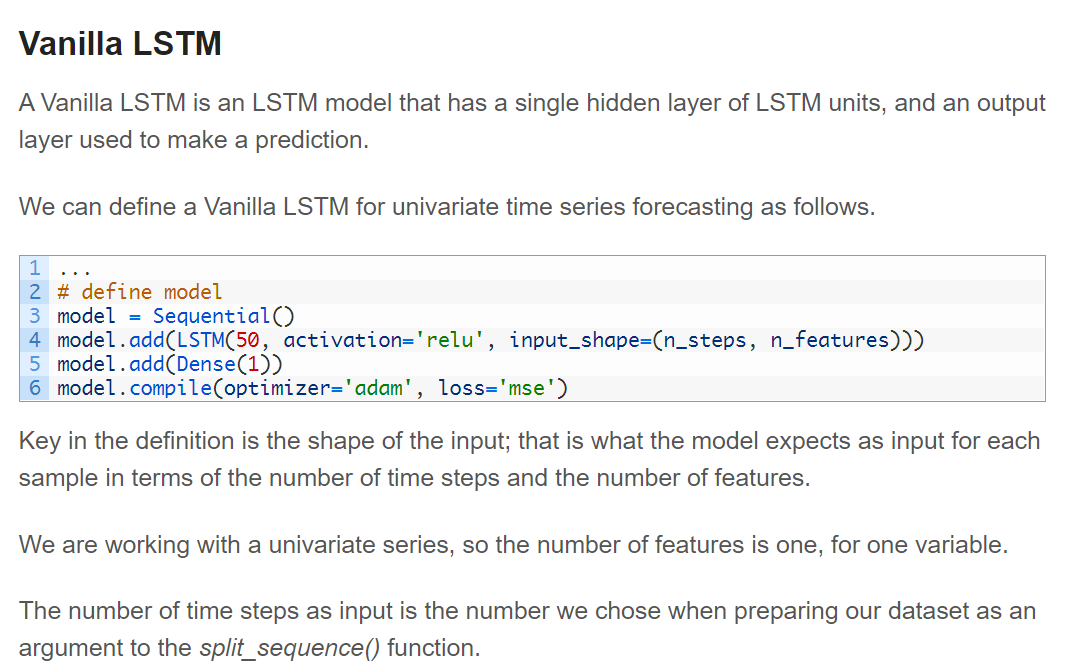
In essence, the layer will contain **multiple parallel LSTM units**, structurally identical but each eventually **"learning to remember" some different thing.**

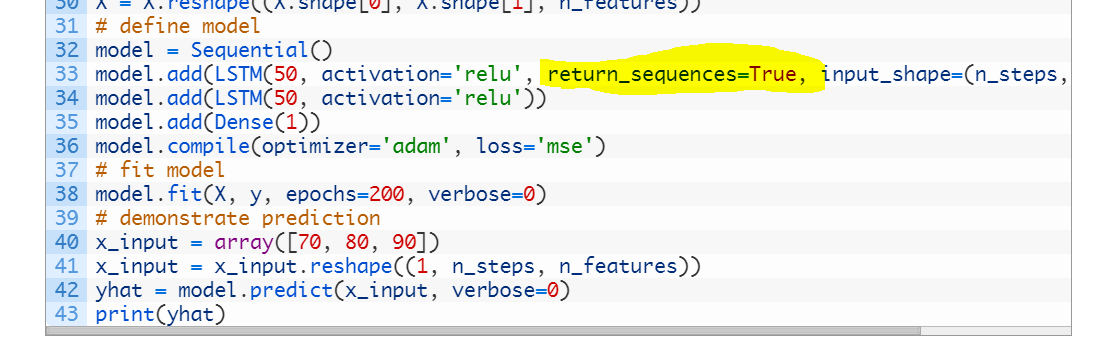
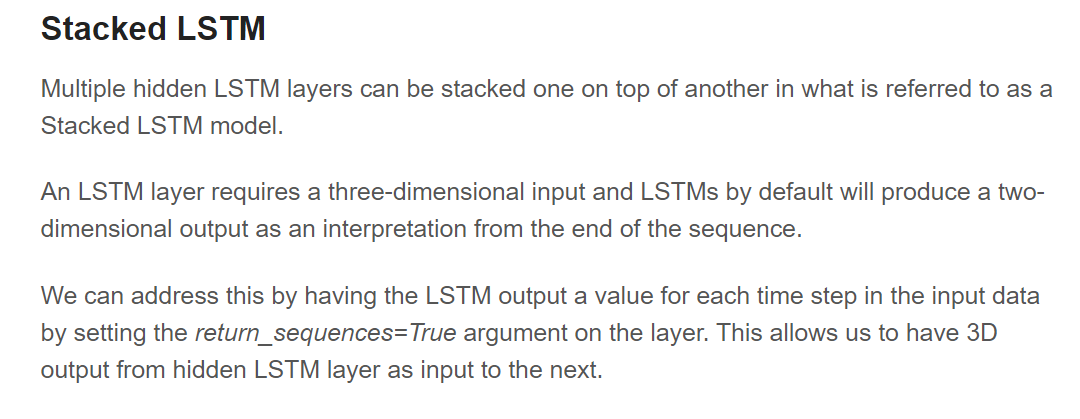
Most LSTM/RNN diagrams just show the hidden cells but never the units of those cells.

Hence, the confusion. Each hidden layer has hidden cells, as many as the number of time steps.

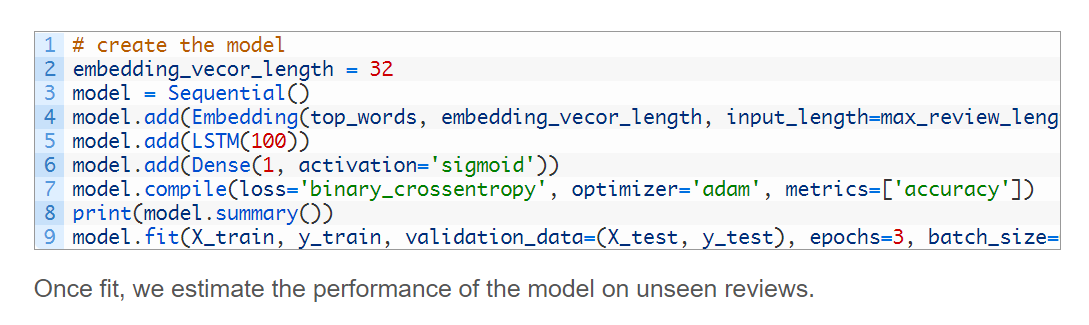
And further, each hidden cell is made up of multiple hidden units, like in the diagram below. Therefore, the dimensionality of a hidden layer matrix in RNN is (number of time steps, number of hidden units).







**For Classification**



**Sigmoid for binary**

**Soft-max for multiclass problem**

**Binary cross entropy**

**Categorical cross entropy**