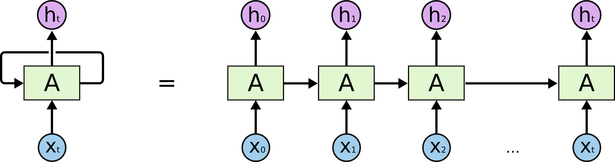
**Assignment 03**

**1. Explain the basic architecture of RNN cell.**

**Ans:** The fundamental feature of a Recurrent Neural Network (RNN) is that the network contains at least one feed-back connection, so the activations can flow round in a loop. That enables the networks to do temporal processing and learn sequences, e.g., perform sequence recognition/reproduction or temporal association/prediction. Recurrent neural network architectures can have many different forms. One common type consists of a standard Multi-Layer Perceptron (MLP) plus added loops. These can exploit the powerful non-linear mapping capabilities of the MLP, and also have some form of memory. Others have more uniform structures, potentially with every neuron connected to all the others, and may also have stochastic activation functions. For simple architectures and deterministic activation functions, learning can be achieved using similar gradient descent procedures to those leading to the back-propagation algorithm for feed-forward networks.

In sequential tasks such as natural language and speech processing, there is always dependence of present input data upon the previous applied inputs. Task of RNNs is to find the relationship between current input and the previous applied inputs. In theory RNNs can make use of information sequence of any arbitrarily length, but in practice they are limited to looking back only a few steps.



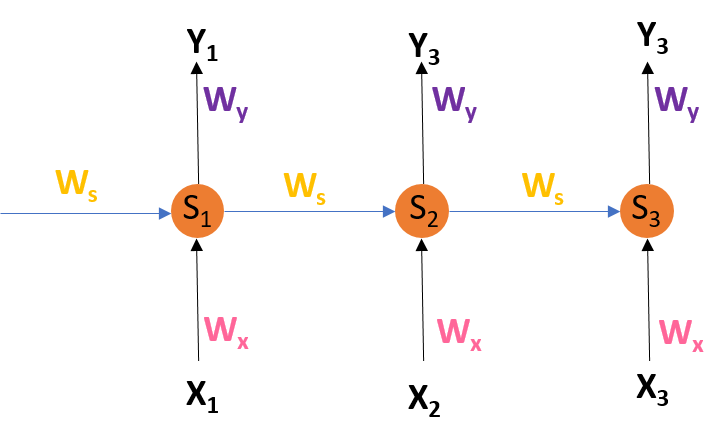
The above figure shows a RNN being unfolded into a full network. By unfolding we simply mean that we are repeating the same layer structure of network for the complete sequence. Xt is the input at time step t. Xt is a vector of any size N. A is the hidden state at time step t. It’s the “memory” of the network. It is calculated based on the previous hidden state and the input at the current step.

Represented by At= f (W Xt +U At-1)

Here W and U are weights for input and previous state value input. And f is the non-linearity applied to the sum to generate final cell state.

**2. Explain Backpropagation through time (BPTT)**

**Ans:** Backpropagation through time (BPTT) is a gradient-based technique for training certain types of recurrent neural networks. It can be used to train Elman networks. ecurrent Neural Networks are those networks that deal with sequential data. They predict outputs using not only the current inputs but also by taking into consideration those that occurred before it. In other words, the current output depends on current output as well as a memory element (which takes into account the past inputs).  
For training such networks, we use good old backpropagation but with a slight twist. We don’t independently train the system at a specific time *“t”*. We train it at a specific time *“t”* as well as all that has happened before time *“t”* like t-1, t-2, t-3. Consider the following representation of a RNN:



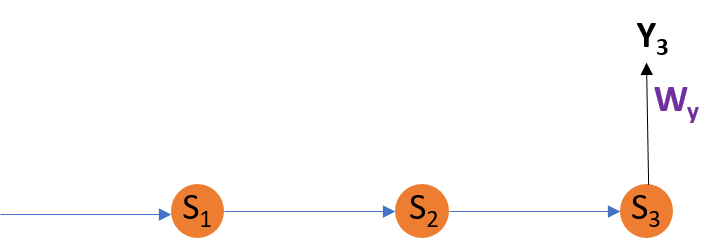
RNN Architecture

S1, S2, S3 are the hidden states or memory units at time t1, t2, t3 respectively, and Ws is the weight matrix associated with it.  
X1, X2, X3 are the inputs at time t1, t2, t3 respectively, and Wx is the weight matrix associated with it.  
Y1, Y2, Y3 are the outputs at time t1, t2, t3 respectively, and Wy is the weight matrix associated with it.  
For any time, t, we have the following two equations:

where g1 and g2 are activation functions.  
Let us now perform back propagation at time t = 3.  
Let the error function be:

, so at t =3,

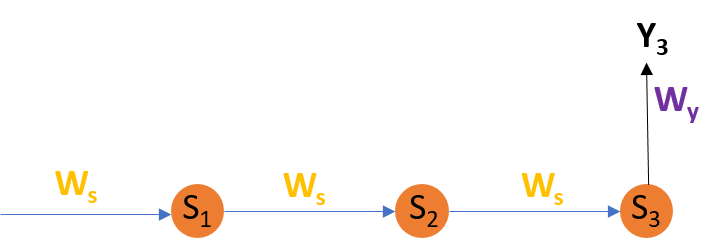
\*We are using the squared error here, where d3 is the desired output at time t = 3.  
To perform back propagation, we have to adjust the weights associated with inputs, the memory units and the outputs.  
Adjusting Wy  
For better understanding, let us consider the following representation:



Adjusting Wy

Explanation:  
E3 is a function of Y3. Hence, we differentiate E3 w.r.t Y3.  
Y3 is a function of WY. Hence, we differentiate Y3 w.r.t WY.

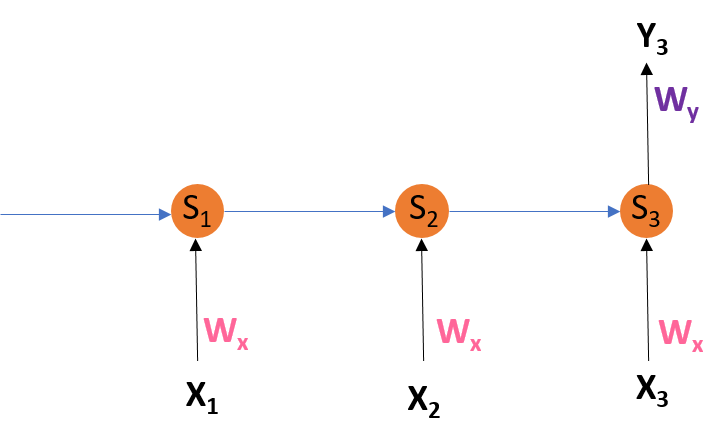
Adjusting Ws  
For better understanding, let us consider the following representation:



Adjusting Ws

Explanation:  
E3 is a function of Y3. Hence, we differentiate E3 w.r.t Y3.  
Y3 is a function of S3. Hence, we differentiate Y3 w.r.t S3.  
S3 is a function of WS. Hence, we differentiate S3 w.r.t WS.  
But we can’t stop with this; we also have to take into consideration, the previous time steps. So, we differentiate (partially) the Error function with respect to memory units S2 as well as S1 taking into consideration the weight matrix WS.  
We have to keep in mind that a memory unit, say St is a function of its previous memory unit St-1.  
Hence, we differentiate S3 with S2 and S2 with S1.

Adjusting WX:  
For better understanding, let us consider the following representation:



Explanation:  
E3 is a function of Y3. Hence, we differentiate E3 w.r.t Y3.  
Y3 is a function of S3. Hence, we differentiate Y3 w.r.t S3.  
S3 is a function of WX. Hence, we differentiate S3 w.r.t WX.  
Again we can’t stop with this; we also have to take into consideration, the previous time steps. So, we differentiate (partially) the Error function with respect to memory units S2 as well as S1 taking into consideration the weight matrix WX.

Limitations:  
This method of Back Propagation through time (BPTT) can be used up to a limited number of time steps like 8 or 10. If we back propagate further, the gradient  becomes too small. This problem is called the “Vanishing gradient” problem. The problem is that the contribution of information decays geometrically over time. So, if the number of time steps is >10 (Let’s say), that information will effectively be discarded.

**3. Explain Vanishing and exploding gradients**

**Ans:** The following is the difference between vanishing and Exploding gradients

**Gradient:** The Gradient is nothing but a derivative of loss function with respect to the weights. It is used to updates the weights to minimize the loss function during the back propagation in neural networks.

**Vanishing Gradient:** Vanishing Gradient occurs when the derivative or slope will get smaller and smaller as we go backward with every layer during backpropagation.

When weights update is very small or exponential small, the training time takes too much longer, and in the worst case, this may completely stop the neural network training.

A vanishing Gradient problem occurs with the sigmoid and tanh activation function because the derivatives of the sigmoid and tanh activation functions are between 0 to 0.25 and 0–1. Therefore, the updated weight values are small, and the new weight values are very similar to the old weight values. This leads to Vanishing Gradient problem. We can avoid this problem using the ReLU activation function because the gradient is 0 for negatives and zero input, and 1 for positive input.

**Exploding gradient:** Exploding gradient occurs when the derivatives or slope will get larger and larger as we go backward with every layer during backpropagation. This situation is the exact opposite of the vanishing gradients.

This problem happens because of weights, not because of the activation function. Due to high weight values, the derivatives will also higher so that the new weight varies a lot to the older weight, and the gradient will never converge. So it may result in oscillating around minima and never come to a global minima point.

**4. Explain Long short-term memory (LSTM)**

**Ans:** Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

This is a behavior required in complex problem domains like machine translation, speech recognition, and more.

LSTMs are a complex area of deep learning. It can be hard to get your hands around what LSTMs are, and how terms like bidirectional and sequence-to-sequence relate to the field.

**5. Explain Gated recurrent unit (GRU)**

**Ans:** A Gated Recurrent Unit (GRU) is part of a specific model of recurrent neural network that intends to use connections through a sequence of nodes to perform machine learning tasks associated with memory and clustering, for instance, in speech recognition. Gated recurrent units help to adjust neural network input weights to solve the vanishing gradient problem that is a commo n issue with recurrent neural networks. The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state. The output of the GRU is calculated based on the updated hidden state.

The equations used to calculate the reset gate, update gate, and hidden state of a GRU are as follows:Working of a Gated Recurrent Unit:

* Take input the current input and the previous hidden state as vectors.
* Calculate the values of the three different gates by following the steps given below:-
  1. For each gate, calculate the parameterized current input and previously hidden state vectors by performing element-wise multiplication (Hadamard Product) between the concerned vector and the respective weights for each gate.
  2. Apply the respective activation function for each gate element-wise on the parameterized vectors. Below given is the list of the gates with the activation function to be applied for the gate.

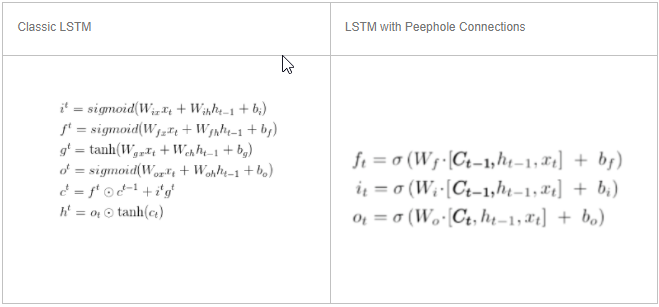
A picture containing screenshot, diagram, clipart, cartoon

Description automatically generated

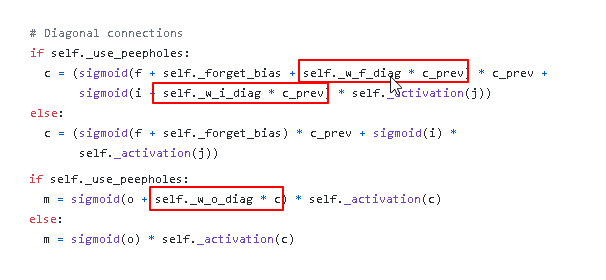
**6. Explain Peephole LSTM**

**Ans:** LSTM peephole conncections is one of improvements for classic LSTM network.

* Difference between LSTM peephole conncections and classic LSTM
* We should compare them with their formulas.



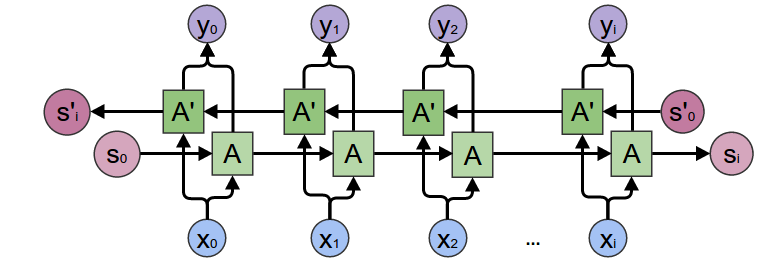
* We can find the main differences between classic LSTM and LSTM with peephole connections are in three gates.
* LSTM with peephole connections add hidden state Ct to three gates in classic lstm.
* We also can find the detail in tensorflow source code.



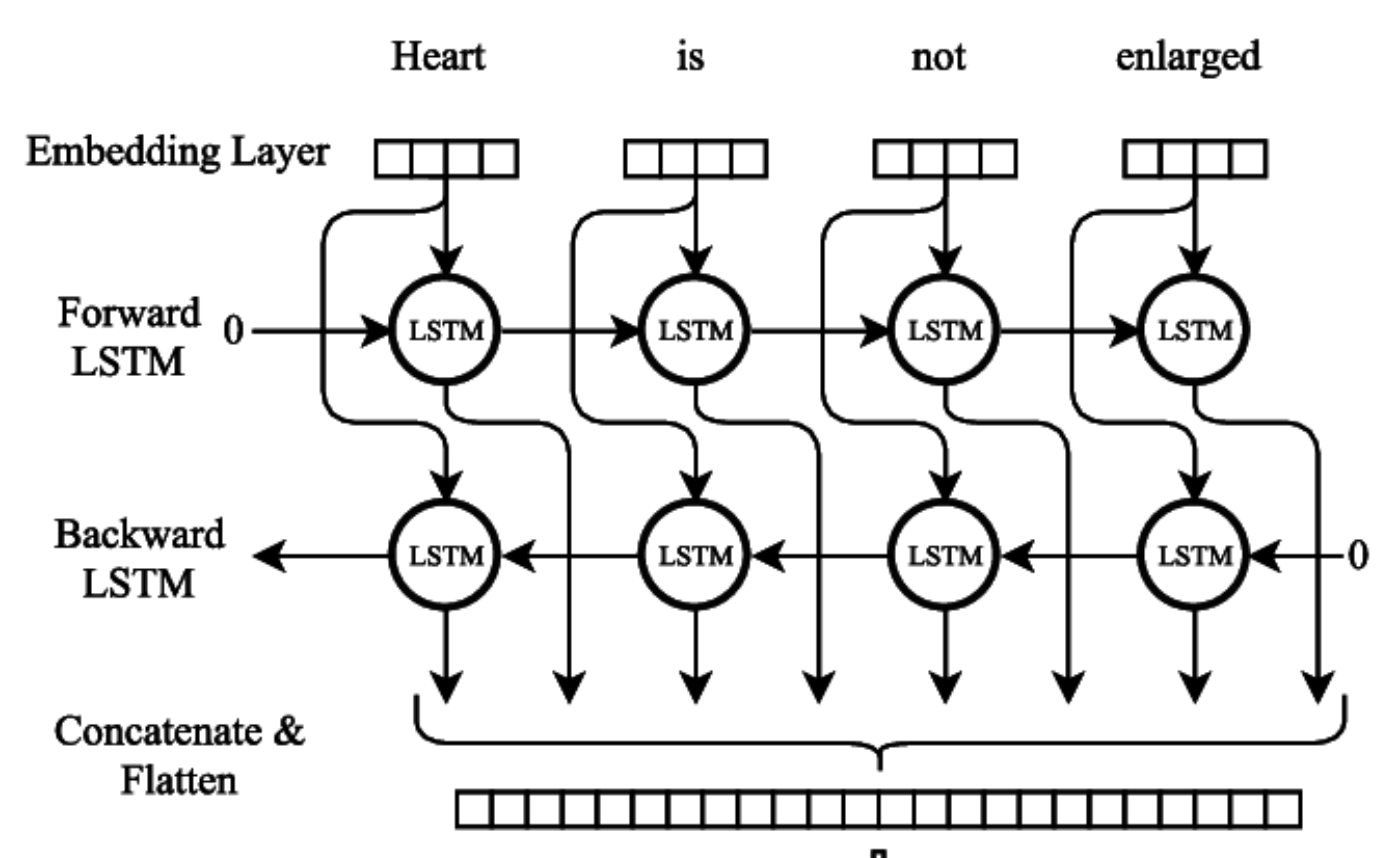
**7. Bidirectional RNNs**

**Ans:** Bidirectional recurrent neural networks(RNN) are really just putting two independent RNNs together. The input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation.

This structure allows the networks to have both backward and forward information about the sequence at every time step. The concept seems easy enough. But when it comes to actually implementing a neural network which utilizes bidirectional structure, confusion arises…



**8. Explain BiLSTM**

**Ans:** A Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction. BiLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm (e.g. knowing what words immediately follow and precede a word in a sentence). 

**9. Explain BiGRU**

**Ans:** A Bidirectional GRU, or BiGRU, is a sequence processing model that consists of two GRUs. one taking the input in a forward direction, and the other in a backwards direction. It is a bidirectional recurrent neural network with only the input and forget gates.

