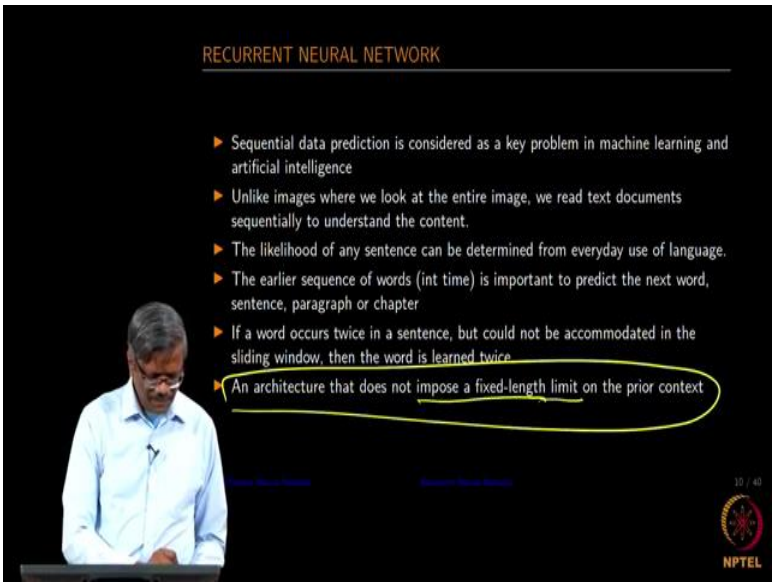
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

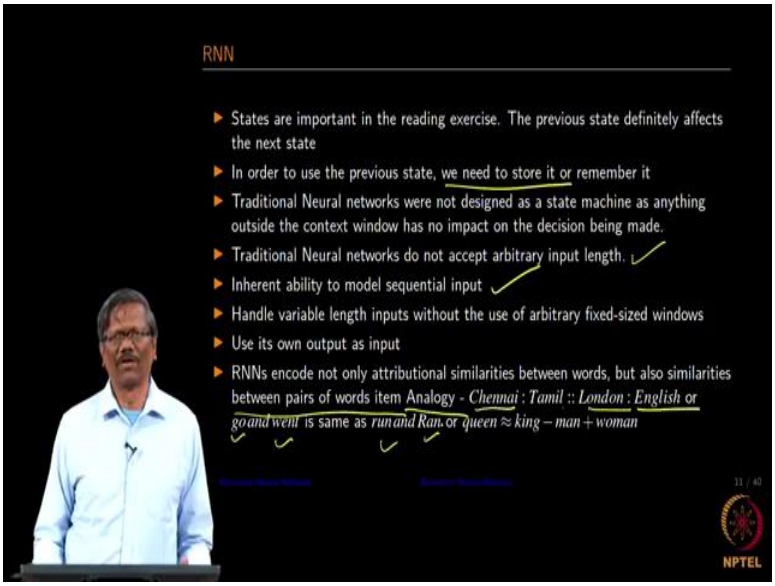
Lecture - 54
Introduction to Recurrent Neural Network

(Refer Slide Time: 00:15)



So, in the 10th slide we going to just look at what I have not covered and then maybe take that. So, this is the key aspect of this. So, we want to create an architecture that does not impose a fixed-length limit all right.

(Refer Slide Time: 00:41)



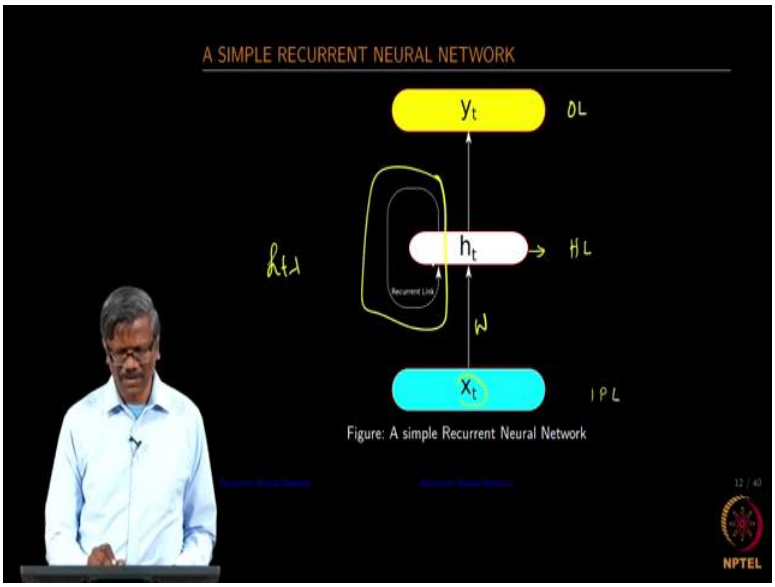
So, what do we require to build such architecture? So, state and so, for now we have never spoken about the word state right. So, now, we are going to be talking about the state; states are very important.

So, for you to really go to the next state I need to know what happened in the previous state. So, without the knowledge about the previous one I cannot move forward ok. So, states are extremely important in the RNN also in the NLP as well right. In order to use the previous state we need to somehow store it right.

So, we need a storage mechanism where these states are stored all right. So, this we already have spoken. So, we should be able to provide a sequential input so, that the model should be able to really take the sequential input of any length all right.

So, we also would see later that the RNNs do not just encode these similarities in between words, but also between the pair of words, it also creates the similarity and also finds out the analogy. For example, if you give the word Chennai is Tamil is the same as London an English right. So, in this case go and went is the same as run and run. So, we should be able to find the analogies in the recurrent neural network as well. So, we will spend one half an hour or 45 minutes session on how to really capture the analogies in the corpus.

(Refer Slide Time: 02:48)

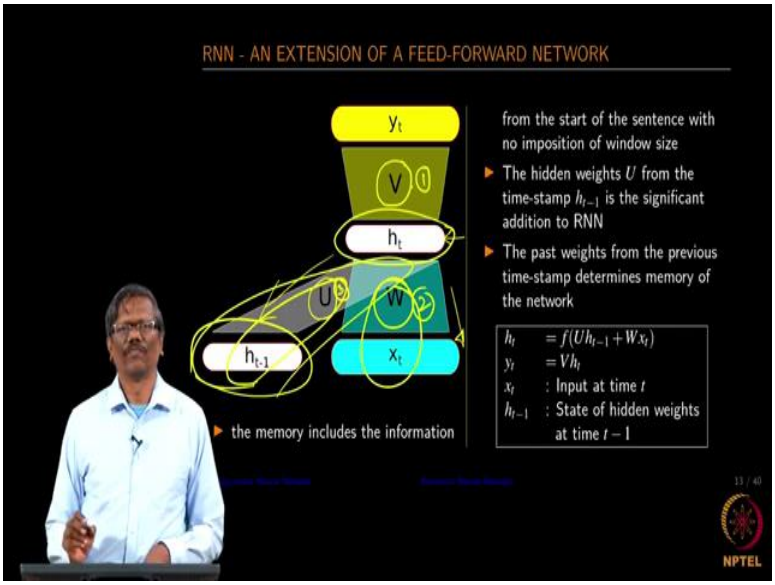


So, as I mentioned earlier, it is almost similar to what we have seen, but there is one small change that you will see here right this is a recurrent part.

So, there is a small loop in the hidden layer part. So, you know this is your input layer, your hidden layer this is your output layer correct. So, we have introduced some small states. So, why we have introduced it here? So, when you take the input and then create a linear combination and then store it as a hidden value right. So, it really takes the essence of the input and the weight vectors that you have trained so, far and that we want to maintain that. So, we captured that in the previous state and then include this as part of the new state.

So, that means we are incorporating whatever we have learned earlier in the embedding layer and then incorporate that as part of the current state. So, we will see how this could be done. So, this is one simple recurrent neural network where there is a small loop introduced in the hidden layer. So, this is going to give us the time series ok. So, this is going to give us the time series let see how it does.

(Refer Slide Time: 04:35)



So, in this slide we will see that the same neural network that I showed you earlier can be unrolled; that means, I have the state of the previously hidden layer and I connect to this state of the current hidden layer through another weight that I have created.

Ok. So, now, we are introducing one more parameter. So, earlier we had the embedding weights, we had the context weights. So, now, we have one hidden layer to the hidden layer there is one more parameter. So, now, if you want to learn; so, we need to learn not only this but also this sorry this would be 3. So, these three parameters should be learned.

So, this really encodes what you have given as an input and combines that with the weight vector right. So, this is the linear combination of these two and this is what is stored as the previous state as well and then the state of the linear combination that we had gotten earlier is connected to the current state through the weights which connect the previous timestamp of hand the current timestamp of h through the weights here all right.

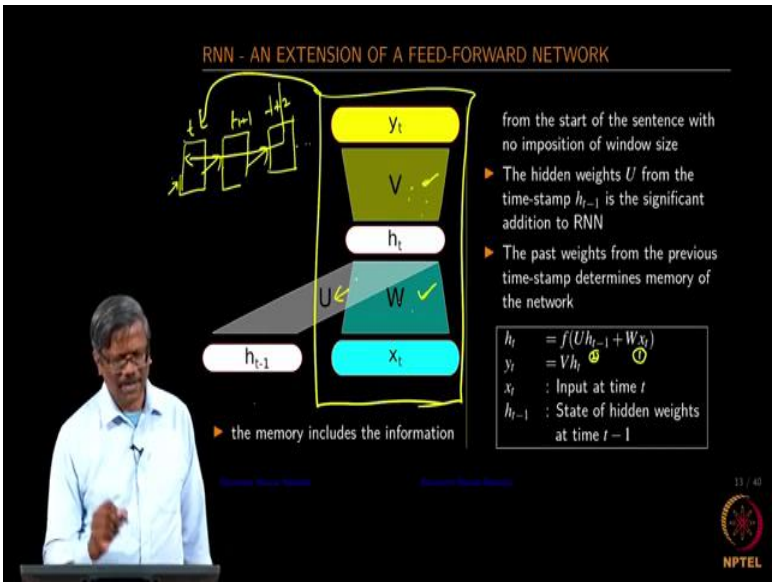
So, this is the memory part that we are talking about, we want to retain this so, that we keep knowing what has happened in the previous time slides ok. Time slides even before that time before that, say it from the start till the time equal to t we will have to capture this and then maintain this in our memory right; that means, it has managed to maintain the activation values of the time slides, activation values of h across the time slides that we are having all right.

So, the updation of weights is very similar to what we had seen the only change that we have is, this activation is calculated by combining these two and this. So, let me erase those so, that become clearer.

So, this activation is computed using 1 and 2 ok. So, this is what we have like two in this ok. So, and then the rest is the same as we had seen.

So, this activation based on the sigmoidal that you have used would be connected to the context weights and then finally, you get an output layer, you apply softmax or hierarchical softmax depending on how you want to optimize your network and then start doing the process of training and so, on. So, when you do the training, we do not just have this let me again erase all those points.

(Refer Slide Time: 08:31)



The backpropagation is not only on these and these weights, it has to be propagated to these as well. So, this is a little hard when compared to what we had seen earlier and if you have multiple time slices you will have more of this particular one. So, it will be repeated. So, you will have one and there is. Equation os shown below

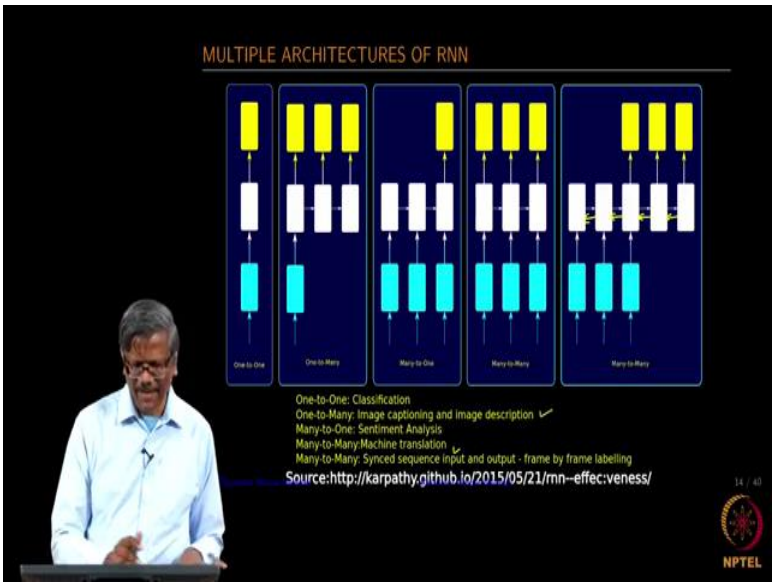
$$h_r = f(u\ h_{t-1} + w_{x1})$$

$$y_1 = Vh_1$$

This is t, t + 1, t + 2 and so, on. So, let us this is my box. So, every time so; that means, the backpropagation is supposed we have only about three of these, it starts from here and then we go back and do it. So, every time when you come from the output layer, you have to update this, you have to update this based on y and V, you have to update this based on Y , V , W and whatever you have earlier and so, on. I am not going to be covering it in this session I will take up separate session to see how training can be handled in the recurrent neural network bond ok.

So, it is just for the understanding of how the weights are calculated. So, we have ht is computed using the function and then y t is computed using V and h t and where h t is your input and h_t -1 is this state of hidden weights at time t - 1 right ok.

(Refer Slide Time: 10:32)



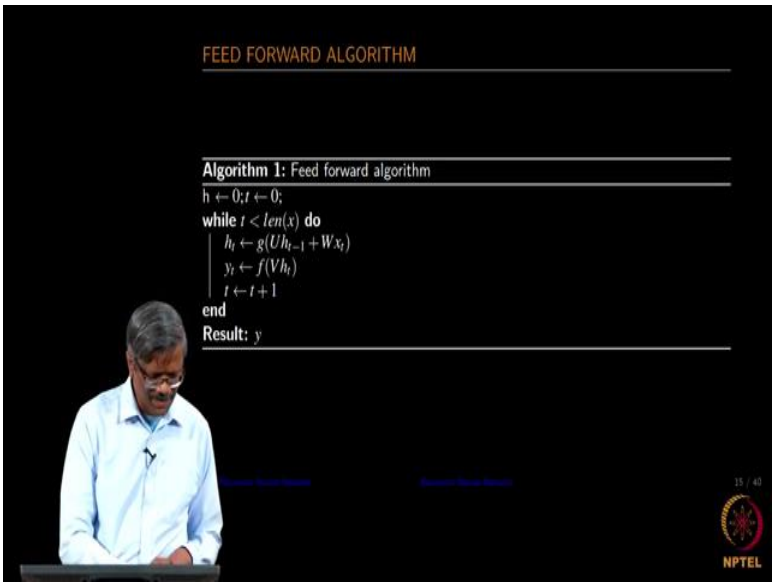
So, there are. So, having understood how a recurrent neural net is constructed right. When you look at the neural net which is a simple one, you have lots of options here unlike in the previous case we can build the neural network using several ways for example, I can just create one to one ok.

So, where I can use that neural net as a standard neural network for classification purposes correct and then I can have one input to many outputs, I can use it for image descriptions. So, I can keep feeding one word and then it can give me various options connected to that particular word and then I can have many to one basically used in the sentiment analysis right. And then we have many to many or we can use it for machine translation.

So, I am talking about this one ok. So, there is an encoding part and there is a decoding part and then there is a many to many combined differently. So, we can use it for frame labeling of video sequences; many people have used this network for various applications I have just listed based on the usage of this architecture for various applications and also you will see that there is some more complex network that we can build, this is another way of creating architecture.

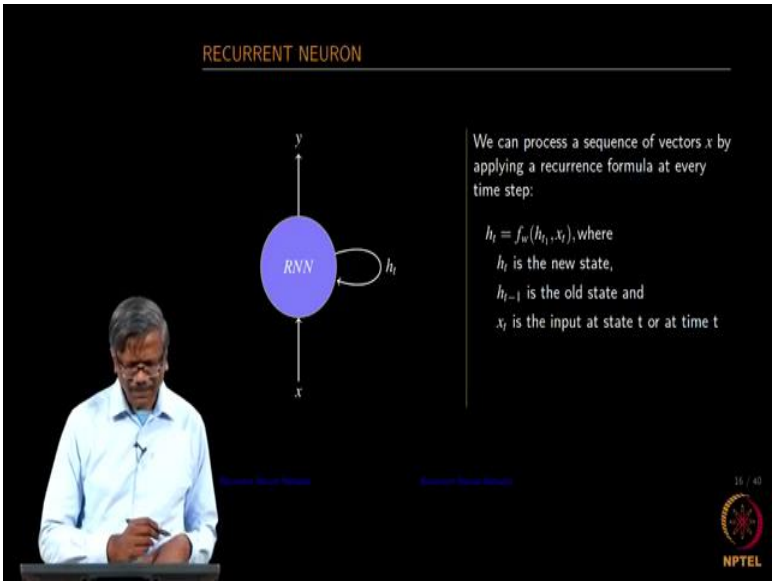
So, there is the enormous capability in terms of change the architecture depending on what you really want to do with this RNN ok. So, this is just to give you an idea of how differently you can create recurrent neural networks.

(Refer Slide Time: 13:00)



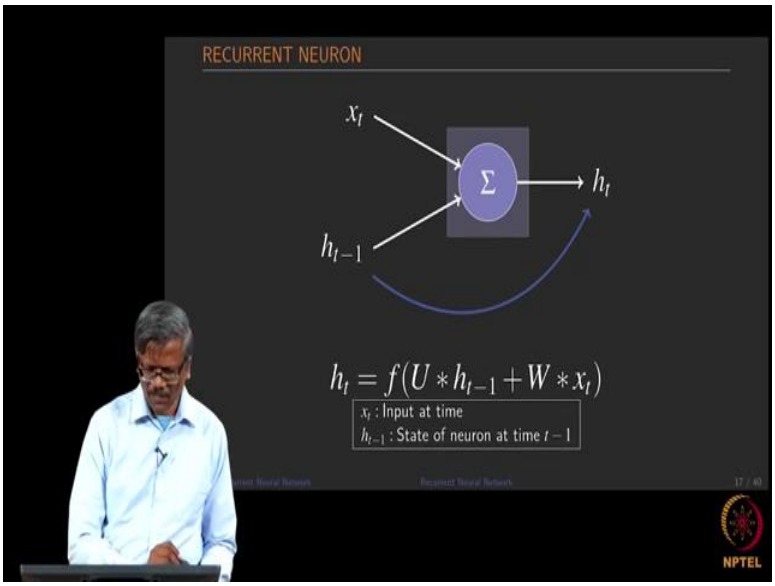
This is a very simple feed-forward algorithm as I discussed this earlier. So, I am going to skip this.

(Refer Slide Time: 13:08)



Again, you will see the various representation of this I just want to give you the various representation of the same recurrent neuron, that appears in various papers research papers and technical papers ok. So, this is one more representation of the same ok.

(Refer Slide Time: 13:32)



So, again the representation of a simple recurrent neuron in a different format is the same as what we saw earlier here right.