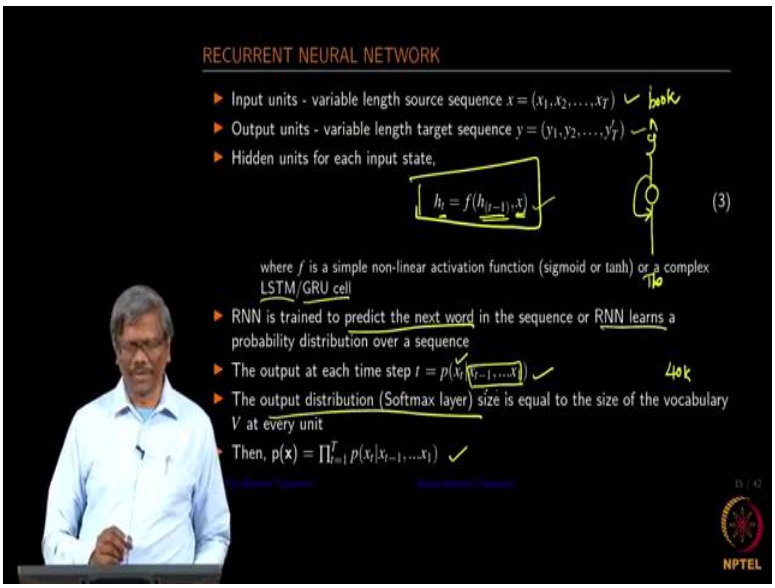
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture -76
RNN Based Machine Translation

(Refer Slide Time: 00:15)



So, as I mentioned earlier we are going to be using the Recurrent Neural Network for this. We know that the input is going to be a sequence, the output is going to be a sequence and ht is computed in this fashion using the previous memory and the new input. So, as I mentioned this hidden unit could be an LSTM or GRU cell. So, every box is trained to predict the next word in the sequence and then it learns the probability distribution.

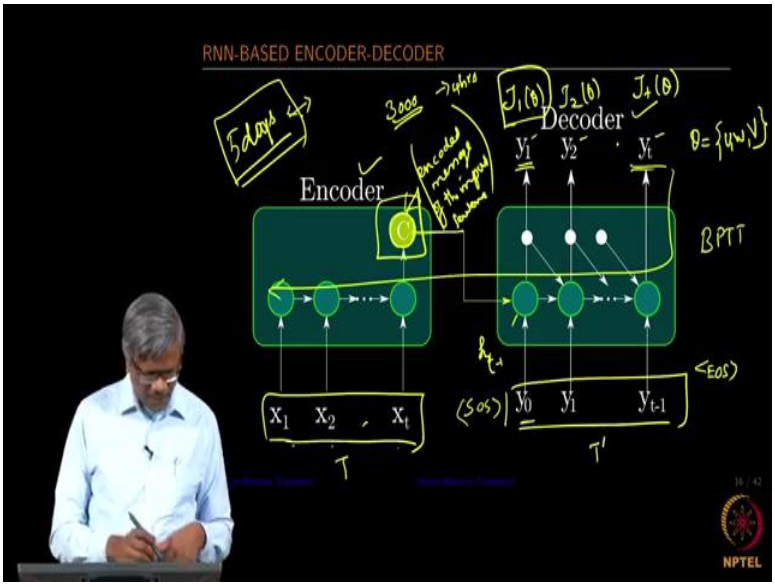
$$P(x) = \prod_{t=1}^{T} P(x_t \mid x_{t-1} \ldots\ldots x_1)$$

For example, given a single-layered one right I will let me use this as y hat; this is our RN. So, if I input let say the here we expect the output as a book right. So, every time we are learning the next word in that sequence and it remembers the previously hidden value using this matrix and then output at each time is again a probability. It is I am bringing in what we had done in the language model; the probability-based language models.

So, it depends on the previous words right. So, the word at time t depends on the words it had seen earlier right, it is conditioned on this a word that it had seen in the earlier time-space. So, the output is a softmax layer. So, if you have 40000 as your vocabulary; you the distribution is going to have 40000 values this also we understand; well I guess right.

And then the probability of the sentence is the computation of the probability of each of the words that we have identified ok. So, this is what exactly we are computing in each box ok.

(Refer Slide Time: 02:37)



So, little more into the diagram side; rather than looking at the sentences; so we have at an encoder, we have a decoder. So, we are going to be inputting x_1, x_2, x t and we going to be outputting y1, y_2, and y t and this time slice is different, this time slice is different ok. So, in this case what we do is we compute the context vector at the end of the encoder computations right; so we are going to be feeding that.

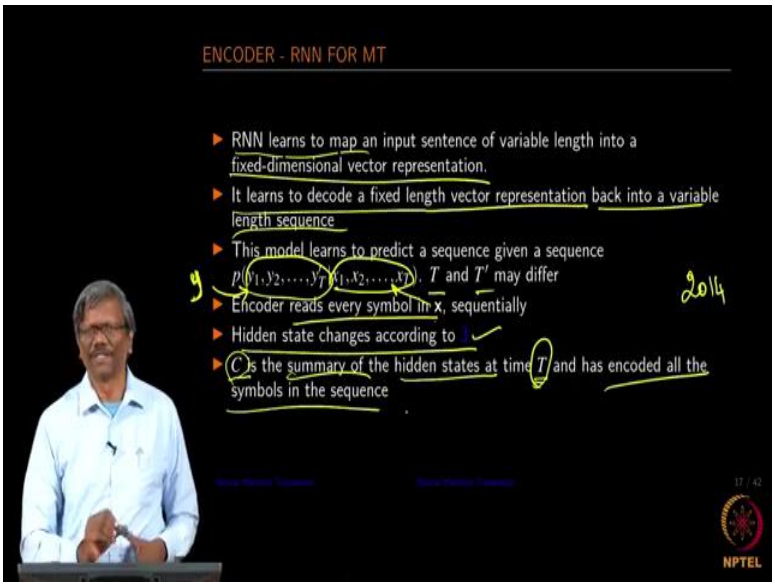
So, this is nothing, but the encoded message like we can call it; this is the encoded message of the input sentence, it contains some numbers; we are going to be using that as an input as well. So, in this case, again there is an h_{t-1} that comes in right. And then there is a context vector that is coming in and then there is an input that is coming into the decoder rights.

We know that these two phrases well before handwrite that the parallel corpora that we have. So, when we input the context into this and then the input word of this and we are going to be getting any 1. So, we know that we have to expect y_1 at this and so and so forth. So, it goes on until y t; so this is a very simple model.

$$h_t = f(h_{t-1}, y_{t-1}, c)$$

So, once the end of the sentence is deducted, so there will be a separate end of the sentence and the start of the sentence which I am not showing here; when that is deducted we know that it is fine for the backpropagation. So, then it comes back and then trains all the weight vectors along the path it passes through ok. So, this is a very simple model for the translation ok.

(Refer Slide Time: 05:12)



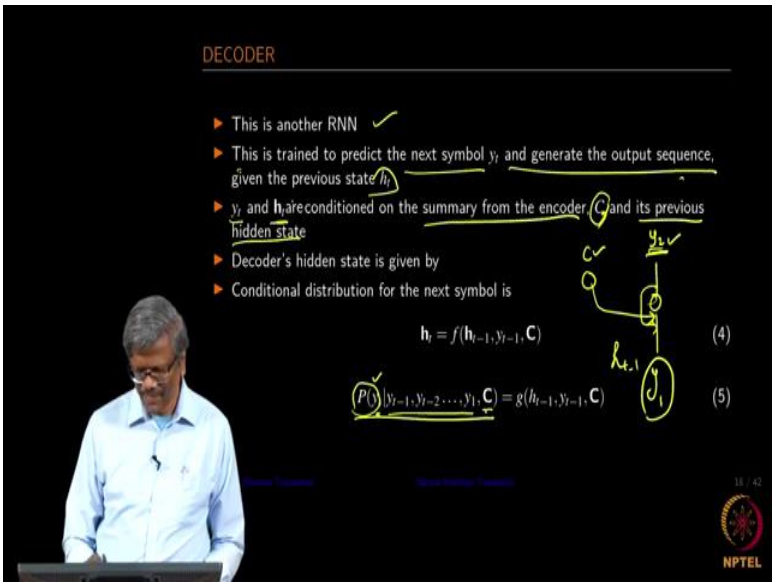
So, this is the first one that came into the world in 2014. So, the only interesting aspect that we have is the context vector that we have created. So, encoder creates the context vector which has encoded all the important things or all the; words in it as a vector and it is fed as an input to this ok.

So, it learned to map the input sentence of variable length into the fixed dimensional vector vs we have seen that. It launched to decode a fixed-length representation back into the variable equal. Then the model learns to predict the sequence given the this is given the entire sequence it learns to predict the entire sequence ok.

And then the time plays as the mentioned would differ. As I mentioned as part of the RNN model every symbol is read sequentially. So, x is a sequence containing x 1, x 2 to x t x 3, and y are in bold represent this in every time one word is read at a time. Hidden state changes according to this equation ok; we know this as well right. Based on our experience in the RNN we know that the hidden chain hidden values at time t depends on the previous memory state, as well as the new input that is coming in ok.

So, we change the hidden state according to equation 3 and then see the summary of a hidden state at time T; at the end of the encoding session we have the summary available as C so that encodes all the symbols in the sequence all right.

(Refer Slide Time: 07:28)



So, when going to the decoder part when we saw in that box right; there is another RNN, this is trying to predict the next symbol as in the encoder as well. And it generates an output sequence. So, in the decoder part; in the encoder part we generate context vector or a summary vector whereas in this case we are generating an output sequence ok.

$$h_t = f(h_{t-1}, y_{t-1}, c)$$

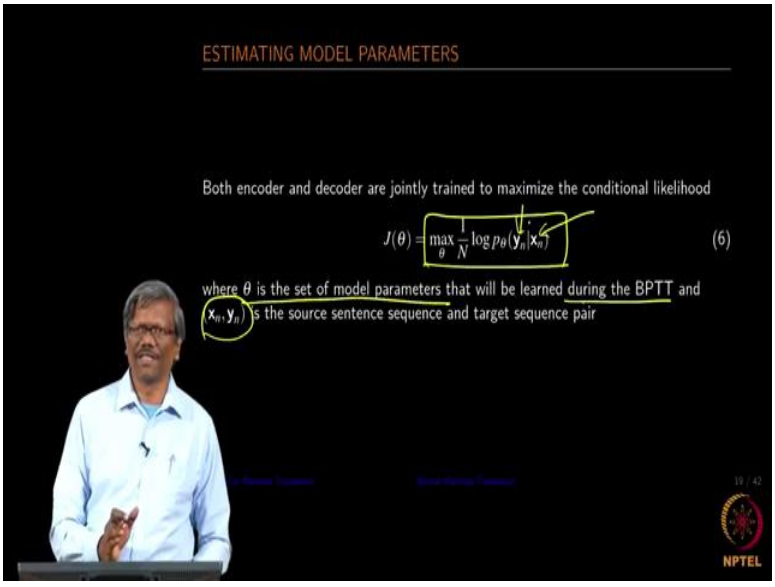
And it depends on the previous state that we had seen earlier and the output is conditioned as I mentioned earlier; the output around the hidden state or the memory is conditioned on the summary from the encoder that is what we have used as C as our

symbol and the previous hidden state. So, this is let us assume that is our decoder and then we have the context vector that is coming in right.

So, here is our y 1 and this is what we have to predict; this C comes in from the encoder part ok. So, this y 2 depends on the context and then the previous hidden state of the decoder and the new input that we provide correct. So, this now has the information about the entire sentence that we had used in the encoder and that is what is passed along here ok.

So, what we get again is a conditional distribution; again if you use softmax depending on the vocabulary, you have that many values found in that ok. We what at the end of it what we are estimating is in the decoder part at the time slice t; the value of this is estimated by using the previous sentences and the summary that we have received ok. So, this is this value is conditioned on the previous inputs, as well as the contacts that we have gotten from the encoder statement ok.

(Refer Slide Time: 10:09)



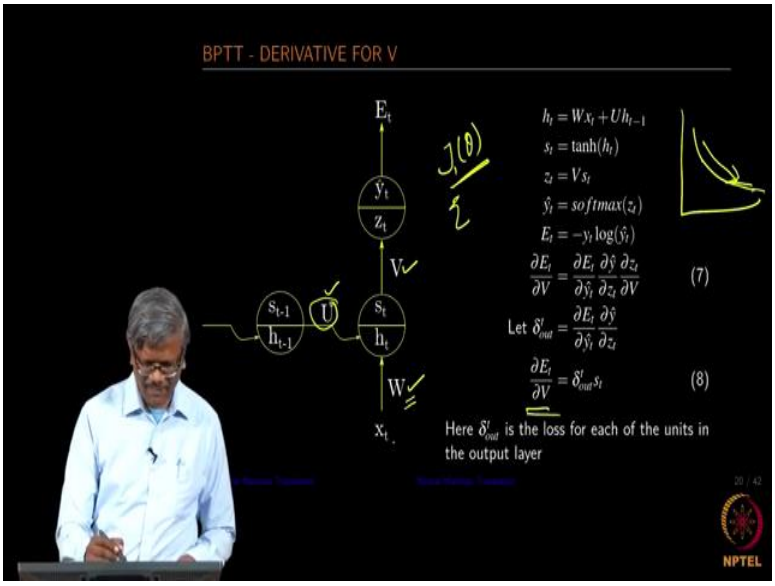
Also next is how do we estimate the model parameters right. So, every time when we do the training part; there is an error that happens. So, we are estimating J_1, J_2, J t. So, theta is the model parameters that we are given to be estimating. Theta is nothing, but the ways that are connecting the output to the hidden layer and then the ways that are connecting the hidden to the hidden way it connecting to the input to the hidden layers

and so on. Supposing if we have a more complex network there will be more parameters that we need to estimate.

$$J(\theta) = \frac{max}{\theta} \frac{1}{N} \log p_0 \left(\frac{y_n}{x_n} \right)$$

So, this J_t is something that we want to estimate right. We want to maximize that, so let us use the log-likelihood and theta is this set of model parameters that will be learned during the backpropagation through time right. And x n and y n are the sequences that we are providing as input and output; this is coming from the encoder and this is what we going to be expecting as this. And then compute error at every stage and then backpropagate and then make the system learn device every time until address minimized ok.

(Refer Slide Time: 11:43)



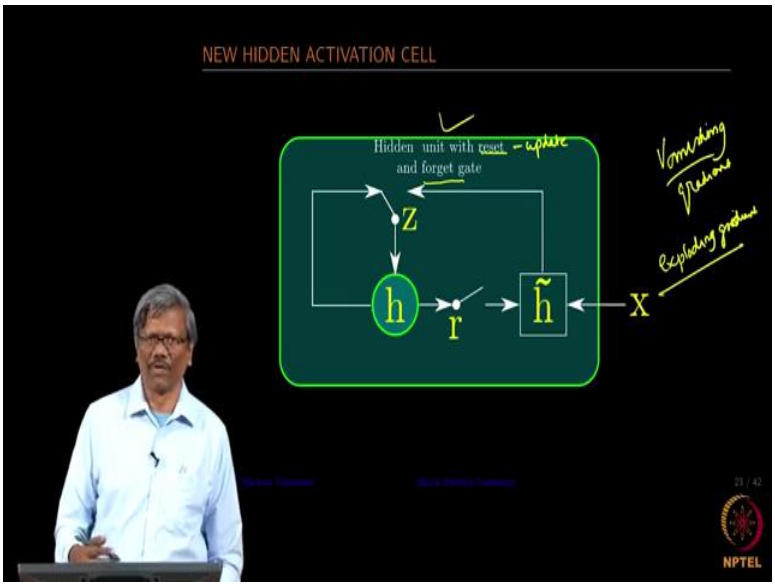
So, this I think again is very familiar to you. So, what I have given here is for your understanding; I have brought some of the backpropagation through time as slides into this. So, as I mentioned what are the parameters that we have to learn; we have to learn V, W and U here.

So, V is the one which is connecting the hidden state to the output unit and then we have the input coming in from the sequence which is connecting the hidden state with the

matrix W and then the previous state is connected using U and then we minimize this and using the partial derivatives and finally, obtain a J value right; we collective value for J ok.

For every output value in the decoder there will be one J which is summed and finally, a core is obtained. We want to use a gradient descent algorithm to find out whether is whether the error is really coming down or not or adjust the input parameters and then initial weight condition and so on to make sure that we follow this path ok. So, once it is minimized we can stop at a certain point and then finally, say that the system has learned.

(Refer Slide Time: 13:22)



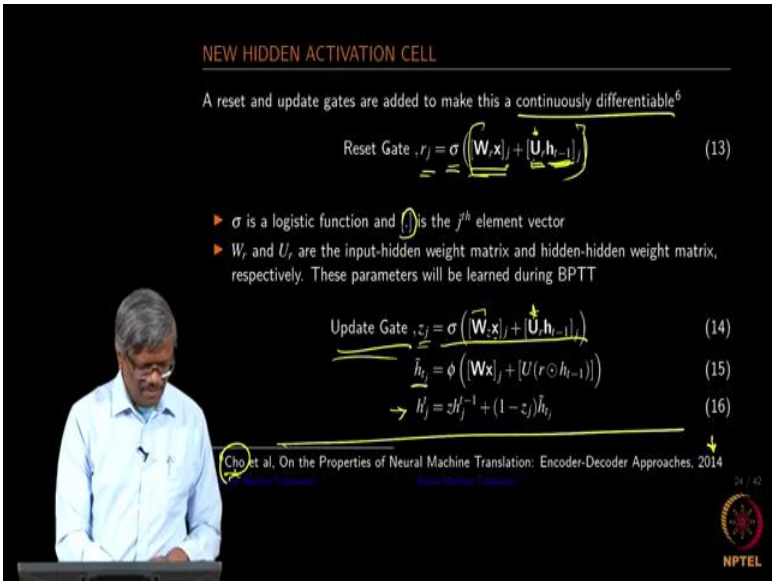
So, in this way we learn V, W, and U and then when we do that we know very well in this normal RNN; we have two problems; one is the vanishing gradient, the second one is the exploding gradient right. So, we want to really avoid this which is very crucial; we can click the gradients to very quickly solve this exploding gradient problem; we had seen all of this earlier. So, what the people who have developed the system had done is created a small cell, very similar to GRU where you have a reset and a forget gate ok.

So, we can call this an update gate. So, in this case you want to maintain that there is a gradient available for you to differentiate when you do the backpropagation through time ok. So, these gates really condition the value in such a way that there is always gradient available for you; they do not really vanish.

One more advantage you know if you say it from the mathematical friend that is what you can say, but if you want to do it from the natural language perspective we want to retain certain important word; even though we have progressed in time you know we do not want to system to forget it. So, we retain some of the important elements using this particular combination of cells that we create as part of the RNN.

So, you can just throw that small h out there and then input this unit and every hidden unit will contain this particular cell. So, only a problem is we add more equations to the system all right.

(Refer Slide Time: 15:35)



So, we have a new hidden activation cell that helps us in terms of solving the vanishing gradient problem by making the values continuously differentiable. So, we have a reset gate you may want to look at the diagram earlier and then see how the reset gate is obtained. So, using the weights connecting the reset gate and the input for the jth element vector; then this is the previous memory connecting the hidden units right.

$$\boldsymbol{r_j} = \boldsymbol{\sigma}\left((w_t\, x_j) + (u_t\, h_{t-1}) \right)$$

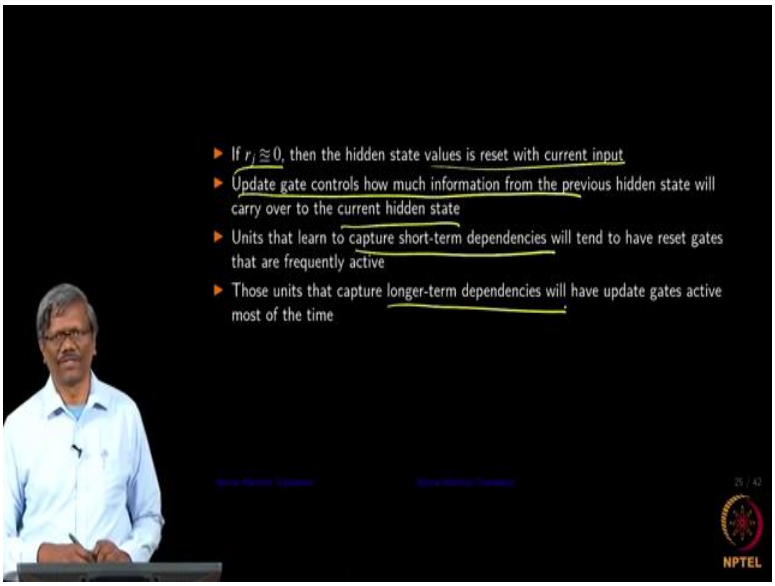
So, this is the weights connecting the hidden units; this is the previous state and then you have the sigmoidal function you get it ok. And then you have a value and this is your j th

element vector; the same way we can get the update gate value like this. Again there is a sigmoidal; there is a weight connecting the input and the update gate. And then again this is the same that we had seen here and a new state is obtained and a new memory is obtained by using this update gate ok.

And finally, the output hidden states are combined using the reset gate and the update gate ok; in this particular presentation I am using this paper for the presentation. So, you may want to go and look at this paper and this is the title and the Properties of the Neural Machine Translation Encoder, Decoder Approaches 2014.

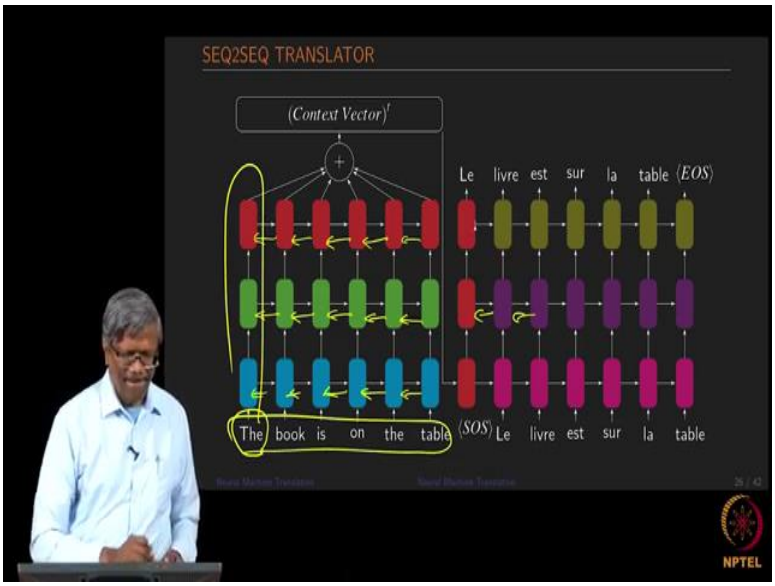
There is also one more paper earlier to this written by the same author that talks about the standard model and think somehow the gate that I mentioned is found as part of the paper written by Cho; that is the first paper that came out in the translation using the neural net model alright ok.

(Refer Slide Time: 17:57)



So, if the reset value is equal to 0; the hidden state value is reset with the current input states. I think we know about this we studied in detail in LSTM as well as and GRU. The update gate controls how much information from the previous hidden state should flow or carry over to the current state. Units; during the train they automatically learn to capture a shot-term dependencies and then they also capture the long term dependencies on its own during the training process ok.

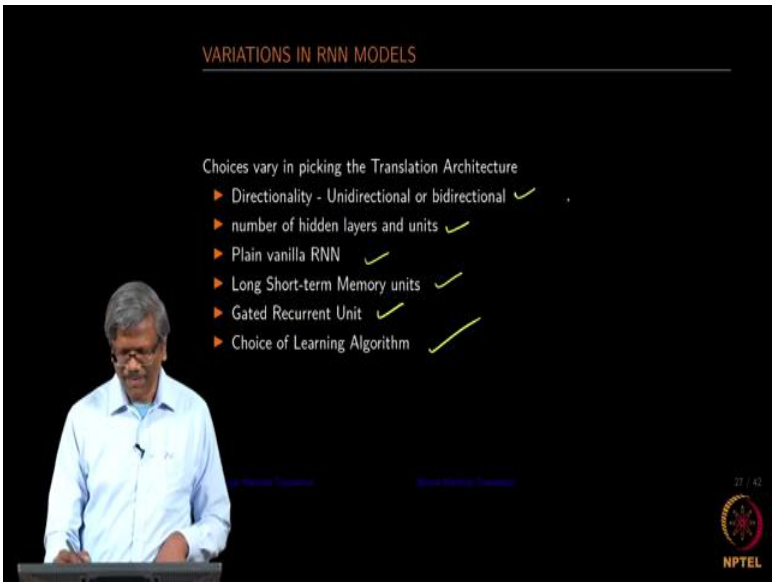
(Refer Slide Time: 18:40)



So, this is one expanded version of the sequence or sequence translator. So, here the English is the input sentence and then the output is a different sentence. So you can have multiple hidden units as mentioned here. You can also have the hidden units traversing back in this fashion; you want to make it more complex so that all the hidden states contain some information about all the states ok.

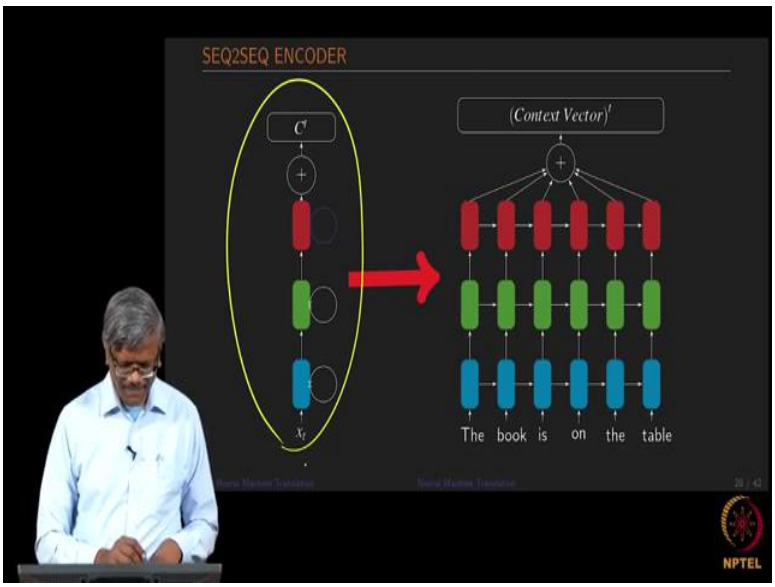
So, when we do this usually the data related to this is found here. So, when we do the backward part of the reverse one, again the whole thing is translated back into this right. So, this is one way of training the network as well. So, again you can use the same model; I am not going to be drawing all of that.

(Refer Slide Time: 19:58)



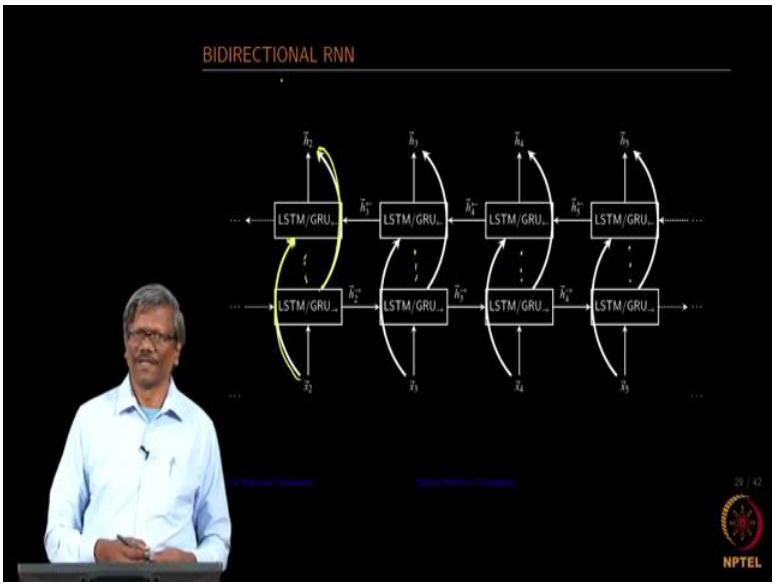
So, there are various variations that you can bring in into this; one using the uni directional or bi-directional as I mentioned earlier. You can bring in more number of hidden layers and units we can use very plain vanilla to do this. We can bring in LSTM as part of the hidden units, we can bring in GRU and you can also have the chance of learning algorithm and you can bring in choice of error mechanism as well as part of that right. So, the permutation and combinations are huge in this case. So, you can have various different combinations of models available for the translation ok.

(Refer Slide Time: 20:39)



So, it can also be represented in one simple way in the rolled fashion.

(Refer Slide Time: 20:51)



Or you can just have values going in this fashion also, you can have multiple of this and pass the input directly into each of those and pass the output of that into various stages rather than sequentially doing it ok.

(Refer Slide Time: 21:17)

The application that we have for this is as I mentioned; you can use this not just for translation, we can use it for summarization. Dialog understanding and then code generation; you can generate C code you can generate python code or we can also

generate some technical papers using this ok. There are a lot of an application people have tried; you may find more than what I have listed here ok.

So, we have described a model where there is two ordinances one is used for encoding another one for decoding. The encoder encodes this sentence into a context vector and then the context vector is fed as an input to the decoder. And then the decoder starts using the values that are coming in from the encoder and from the previous year and state of the decoder and the input values and start creating values for the parameters which we just use as theta.

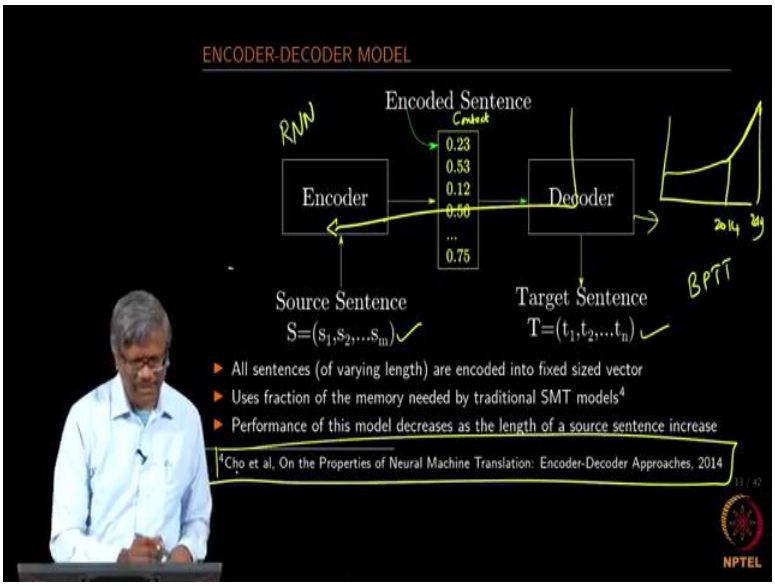
So, there U, W and sorry V ok. And we use another mechanism and find out how different y 1 is from the actual expected output pass the error back using the BPTT and train these models for so many hours to finally, have a translation model which when provided the sequence will give you the output sequence. So, people have used huge corpus or other parallel corpora to do the training.

I think in one case they have tried it for 5 days they train the system for 5 continuous days ok. For a large corpus, I think you cannot avoid this for about 3000 sentences in English and French ok; you took close to about 4 hours on my small computer. So, thinking of a few hundred, thousand, or a million parts of a sentence of 5 days is nothing ok.

So, in the space where we have a lot of computing power neural net really provides the capability in terms of doing a good job in the translation, but you have to be patient with respect to the training. It should be very good in terms of choosing the right input parameters for the initial states ok; that itself is a good research topic and you might find them and various other papers are of machine translation.

In this case we have just use the entire sentence as a context for translation. So, later we will also use another model where the context is broken down into smaller pieces and then how those smaller pieces can really help in terms of doing a better job all right. So, for this session I have used two papers.

(Refer Slide Time: 24:59)



So, one is this and I am not finding the other one maybe you want to search for the author Cho and others and that paper gives you the model that we have just described.