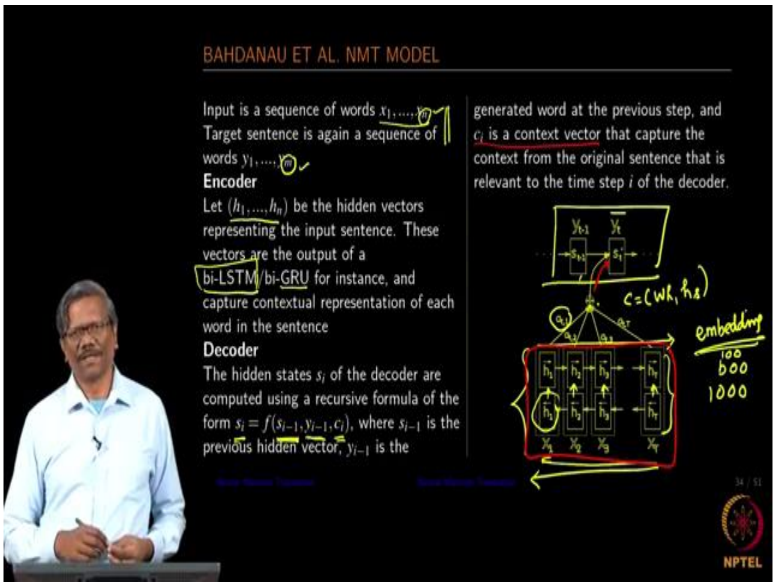
Applied Natural Language Processing
Prof. Ramaseshan Ramachandran
Department of Computer Science and Engineering
Chennai Mathematical Institute, Madras

Lecture – 79
Research Paper discussion on 'Neural machine translation by jointly learning to align and translate"

(Refer Slide Time: 00:15)



BAHDANAU ET AL. NMT MODEL

Input is a sequence of words $x_1, \ldots, x_n$
Target sentence is again a sequence of words $y_1, \ldots, y_m$

Encoder
Let $(h_1, \ldots, h_n)$ be the hidden vectors representing the input sentence. These vectors are the output of a bi-LSTM/bi-GRU for instance, and capture contextual representation of each word in the sentence

Decoder
The hidden states s_i of the decoder are computed using a recursive formula of the form $s_i = f(s_{i-1}, y_{i-1}, c_i)$, where s_{i-1} is the previous hidden vector, y_{i-1} is the generated word at the previous step, and c_i is a context vector that capture the context from the original sentence that is relevant to the time step i of the decoder.

So, let us look at it look at this model at a very high level and then go into the details of that in the next slide. So, as we do in any model let us define the fundamentals. So, input is a sequence of words and then the size is given here that these length of these sentence. Then we have a target sentence which is again a sequence whose length is m. So, we can have as longer sentence producing a shorter translation or a shorter sentence producing a longer translation as well. So, this we need to be keeping in mind right. So, the sentences are not of the same size ok.

Let us define the hidden units ok. So, these are all the vectors that you going to have in the neural net and then this one could be an LSTM or GRU. So, the hidden units could be a cell which is an LSTM cell or a GRU cell which we saw earlier and, then now there is a new one that is coming in it is a bi-directional one right. So, in the bidirectional model what happens is there are layers of hidden units and then one set of hidden units will be learned from the last word to the first word the other one will learn from the first word to the last word right and each one of them is connected this way too ok.

So, this is another model where you can connect these two layers in the upward direction as well. So, one set of units learnt from the last word to the first word other one learns from the first word to the last. So, we know that these hidden units are the representatives of our input. So, connected through the enduring weights right and then these words are for using our embedding meaning that we are not going to be using the one-hot vector and then inputting it. So, we going to be looking at the words from the embedding table that we have created earlier right I am sure you remember that.

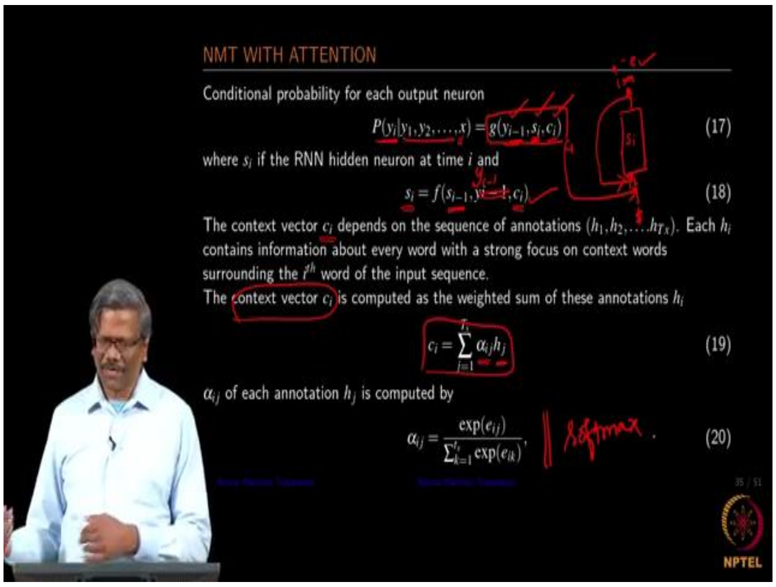
So, embedding is nothing, but the vector that represents the word; not only that word it also represents the contextual information related to the words in the entire vocabulary ok. So, that is what we going to be feeding. Some embeddings sizes would be about 600, some use 1000 or you can also use as low as 100 as the length of the embedding vectors. So, that is what we input. And, h_1 to h_t they represent the input embeddings and then we have to create the alignment which we want to use for the translation right.

So, how do we create the alignments? Let us look at the decoder and the decoder is this and then these are all the hidden states that you compute. These hidden states are computed based on the context that we are going to be getting from the hidden units right. So, we worry about this in the next slide. So, assume that we have computed the attention scores and then we created a weighted sum of the attention score which we call the context. Context is the weighted sum of the h of this input and also we are going to be getting some more values coming in from the decoder ok.

So, that is where we spoke about peeking into the input side right. So, let us use that as I am just using some notation here. So, the context is not just about the weight weighted sum of the hidden units that are coming in from the input side, but also it contains some values from the decoder right, and then that spread into the state again for you to create the next output.

So, now, let us look at the s i; s i is the state of the hidden units in the decoder which is a function of the previous state function of the previous input, and the context that we are creating a that is what is going in here as the. So, let me use a different color. So, this is what is going in as an input to the hidden state in the decoder all right. This c i is our context vector there going to be creating based on the hidden states based on the input coming in from the decoder and that is again fed back into the hidden state of the decoder and then finally, our output is created.

(Refer Slide Time: 06:35)



NMT WITH ATTENTION

Conditional probability for each output neuron

$$P(y_i|y_1, y_2, \ldots, x) = g(y_{i-1}, s_i, c_i) \quad (17)$$

where s_i if the RNN hidden neuron at time i and

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (18)$$

The context vector c_i depends on the sequence of annotations $(h_1, h_2, \ldots h_{T_x})$. Each h_i contains information about every word with a strong focus on context words surrounding the i^{th} word of the input sequence.

The context vector c_i is computed as the weighted sum of these annotations h_i

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \quad (19)$$

α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \quad (20)$$

So, let us look at this first from the equation perspective and then see better diagrams ok. So, what are we going to be doing? So, we going to be conditionally creating the output all right. So, these are a conditional probability. The y i is obtained by using the previous output values given the input sequence ok. So, in it can be written as a function it is a let me just write one here. So, this is our decoder.

$$s_i = (s_{i-1}, y_{i-1}, c_i)$$

$$P(y_1 | y_2, y_3, \ldots\ldots x) = g(y_{i-1}, s_j, c_j)$$

So, we are going to be getting the output let us say I am translating from French to English. This is our start of this sentence I am just marking it as a dolor this is a function of the previous value. This is this start of the symbol is our previous value here and then it is going to be the value that your computing based on the context and the s i ok. So, let us assume that this is your s i. So, we have the c i coming into this and then finally, we have a softmax which will give you this output ok.

$$s_i = f(s_{j-1}, y_{t-1}, c_i)$$

$$c_i = \sum_{j=1}^{\pi} \alpha_{ij} \; h_j$$

$$d_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{t} exp(e_{ik})}$$

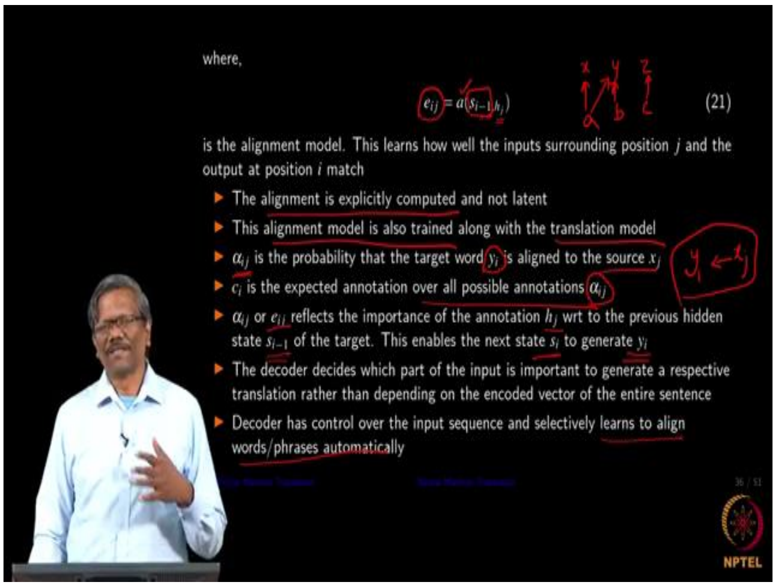
$$c_i = \sum_{k=1}^{T} a_{ij} \; h_j$$

So, this is the standard and network architecture you would have seen correct and this is what we going to be using ok. So, the output word in the decoder depends on the previous word, the hidden state, and the context coming in from the encoder. Then s i is the function of the previous state this should be y i minus 1 the previous output and the context coming in from the encoder right.

And, now we need to find out what is c i. So, actually they coming from the top right. So, we are we found the conditional probability of y i and then for us to find that what do we need we require yi, si and ci. To get s i which is the function of the previous state right. So, let me use this standard model, and then it gets c i from the encoder and then you compute the s i all right. So, this is from the decoder side.

So, how we could compute ci? ci vector is our context that we are inputting into the decoder correct. So, we now have to come to the encoder side and then see how c i could be computed. Remember we saw there are a i's going in right and then this is the weighted sum I told you earlier. It is the weighted sum of the input unit input hidden unit and another value that we would be computing now ok. So, what is alpha states? Alpha i is our annotation for h j which is computed using a softmax. So, this is the softmax right.

(Refer Slide Time: 10:35)



where,

$$\boxed{e_{ij}} = a(\boxed{s_{i-1}}, h_j) \tag{21}$$

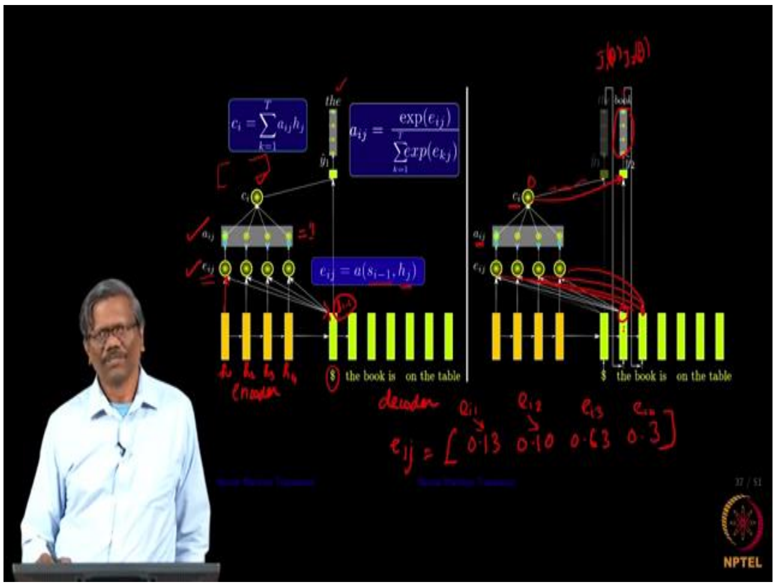
is the alignment model. This learns how well the inputs surrounding position j and the output at position i match

- The alignment is explicitly computed and not latent
- This alignment model is also trained along with the translation model
- α_{ij} is the probability that the target word y_i is aligned to the source x_j
- c_i is the expected annotation over all possible annotations α_{ij}
- α_{ij} or e_{ij} reflects the importance of the annotation h_j wrt to the previous hidden state s_{i-1} of the target. This enables the next state s_i to generate y_i
- The decoder decides which part of the input is important to generate a respective translation rather than depending on the encoded vector of the entire sentence
- Decoder has control over the input sequence and selectively learns to align words/phrases automatically

So, what is e ij? e it is now coming from the decoder, got it? So, this is our decoder hidden value from the previous state and then we are computing e ij using the previous state of the hidden rather previous state of the decoder and the current state of the input or the encoder here right. So, this is our alignment model. So, this is the alignment model that we are computing.

So, now, coming from the bottom to the top so, we let us look at this from here ok. So, here let me remove some of this. So, we are going to be computing e ij here. So, using this e ij is a function of s I and h j right. So, e i is going to be computed using this state and the hidden value there are coming in here. So, whichever state you are this is now we e i1 is computed using h 1 e I 2 would be computed using this and so on. And, then I represent the hidden unit of the decoder ok, is this clear? So, I think a little confusing because they are coming from the top.

(Refer Slide Time: 12:30)



$c_i = \sum_{k=1}^{T} a_{ij} h_j$
$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{kj})}$
$e_{ij} = a(s_{i-1}, h_j)$
the
$\hat{y}_1$
$\hat{y}_2$
book
c_i
a_{ij}
e_{ij}
h_1 h_2 h_3 h_4
encoder
$ the book is on the table
decoder
$e_{ij} = [0.13\ 0.10\ 0.63\ 0.3]$
NPTEL

So, now, let us go to the diagram where the explanation would be a little clearer. So, now, I am actually expanded what we saw in the previous diagram. So, now, we have the encoder and this is our decoder ok. I have just assumed about four units and then the translation is going to be for the set of input the translation going to be the book is on the table ok. So, we going to be computing the first one which is the right.

How do you compute the that is where we started from the original equation and then we computed c it and then we found what was an ij and then what is c ij and so on. So, now, we go from the bottom to the top ok. So, this is our s I minus 1, let us assume that this is our s I minus 1. So, taking the value from here and then value from here right. So, we do a dot product of this e it is nothing, but the dot product of s I minus 1 and h 1, s I minus 1 h 2, s I minus 1 h 3, s I minus 1 h 4. So, we compute a value right.

So, it contains each one we will contain some let me just write it in terms of some numbers I am making them up. So, e it is written as a vector in this form which could be ok. So, each and everyone is an element ok. So, this should be e i 1, e i 2, e i 3, e i 4 and then we take a softmax of this vector. So, once you take this of softmax of this vector, the sum of all those values would be equal to 1 here and you know how a ij is obtained now we know how aij is obtained here using this formula. So, we got this correct ok.

So, we found what how to get this e ij, we now know how to compute a ij which is a softmax of all the values coming in from the e ij values and then we compute the weighted sum of all these a ij values and provide a context vector. So, this now cij is nothing, but the weighted sum of these values and the input stay ok. So, now, we take this and then use this. So, this is one way of doing it. The other way also is to take the value from here and then do it. So, there are multiple ways of actually creating the context one directly taking the value from this and then created creating a weighted sum or you can also do it this way ok.

So, what we have seen so far right. When we compute the values of e ij, a ij, and c i now we know that the decoder is trying to find out how can I really identify who is aligned with me at this particular time slot ok, this or this or that ok. So, it could be this or this could be this word I am sorry, I should be using either this word, second word or third word or fourth. So, it is trying to peek into the input or into the encoder to figure out how to really align with that. During the training process automatically they get adjusted the values or adjusted and we will finally, get the alignment matrix in the right form.

So, once you find a c ij which is nothing, but the summation of all the a ij's and the h ij. So, this is again a vector you feed that into the decoder ok. So, used to compute y t and then there is a softmax that you can compute and finally, you get the output as yeah right. So, here this diagram is a little more elaborate when compared to what we saw earlier and this explains clearly how each value is computed. So, I gave you all those computations on this slide itself all right.

So, when we computed in this form so, how do you go to the next stage ok? So, we have done it for the first word right starting with the first word and then for the second word again we do the same thing. Taking this s i and then computing e ij's and then computing e wij creating a new context vector. So, this context vector that you are creating is with respect to this, correct? And, then use this to it should be this; it should be this. It is fed into the second stage and then later we use the softmax by computing all these values here and then say that this is broken. And, then here we are going to be having a j 1 theta j 2 thetas you know we are computing the error every time right. This is the training process.

And, then third what will happen is you take this one and then start right. So, that means, now the third word is important and it is now trying to peek into the alignment space to find out with a set of words in the input is aligned with this ok. Now, the context is created there will be a new one and the input for this also is coming in from the previous word. So, we keep going and going in this fashion until we find the end of the sentence ok. So, every time we keep computing it ok.

So, there are multiple ways of doing this we will talk about this process of stochastic gradient descent or batch processing or mini-batch at the end of this session ok. So, the idea is to calculate this and then minimize the error so that the weights are learned at the end of the training ok. So, this is what (Refer Time: 20:35) has created. So, now, let us look at the slide again to understand what we have done.

So, here the alignments are explicitly computed that we saw right. So, we are trying to peek in from the decoder into the encoder and then trying to compute the alignment vectors. The alignment model also is train along with the translation model. So, there is no separate model that we are creating like what we have done in the SMT space.

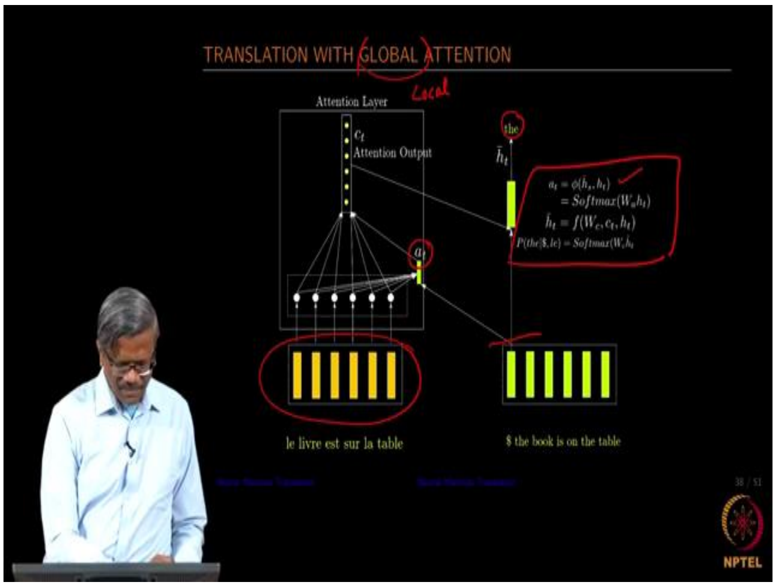
So, alpha ij is the probability that the target word y j is aligned with the source x j I am sorry to align with the target word y and this source; so, that means, we are trying to find y i is aligned with x j you remember the alignment used to have x, y, z – a, b, c right. So, alpha ij is the probability that the target word y i is aligned with the source x j; c i is the expected annotation overall possible annotation that we have. That means when you compute the c i right.

So, it is computed as the weighted sum. So, it is the expected annotation over all possible annotations that you have for that particular time slot; e ij reflects the importance of the annotation history with respect to the previous hidden state ok. So, this enables us to calculate the next state which will in turn generate a predict the next word.

So, during the training the decoder decides which part of the input is important to generate respective translation rather than depending on the encoded vector of the entire sentence right. So, in the model that we saw earlier in the previous session, we need to have the entire context available for you to do the decoding. In this case, now it is not required. It can figure out which part of the sentence or the encoder which part of the encoder is useful and then starts aligning it automatically.

The decoder has control over the input sequence and then selectively learns to align words and phrases automatically. So, again for you to get to this level we need to have a very huge corpus, right then only the patterns would become very clear to the machine. During the machine learning cycle the patterns will start emerging clearly for this system and it is starting to align automatically all right.

(Refer Slide Time: 23:56)


TRANSLATION WITH GLOBAL ATTENTION
Local
Attention Layer
c_t
Attention Output
a_t
$\tilde{h}_t$
the
$a_t = \phi(\bar{h}_s, h_t)$
$= Softmax(W_a h_t)$
$\tilde{h}_t = f(W_c, c_t, h_t)$
$P(the|\$, le) = Softmax(W_s \tilde{h}_t)$
le livre est sur la table
$ the book is on the table
38 / 51
NPTEL

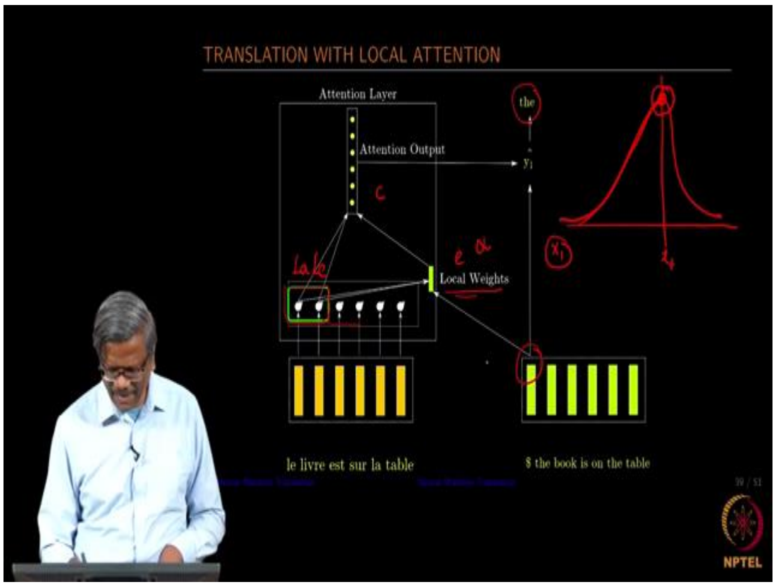
So, what we actually saw was just the attention model where we had looked at the global attention right. So, that means, we looked at every state of the encoder. So, we have not left out anything; that means, we have taken into consideration every state in order to compute the attention score right and that what we had used to compute or predict the next word right.

So, there are variations to this instead of keeping the attention for all the input hidden states so, can we partition it so, that is the next stage right. So, the attention is coming in through this; this is one way, where we capture the attention score for the entire input hidden states initially using the state coming in from the decoder right. So, instead can we just look at some of these not all of the hidden states of the encoder.

So, what we saw earlier was a global model. So, then now we need to see if we can localize that. So, still more control in terms of figure what part of this I need to have. I do not want to really look at every phase of h j that I have in the encoder; I only want to look at only one portion of them. So, there comes the local model were we going to do in the translation with local attention, we will just see in the next one.

So, this computation is very similar actually you will see variations in the equation, but they are almost the same you know some mode write it in this fashion the attention score and then your h t is nothing, but what we saw as s in the previous case and so, on. So, do not get confused, but if you look at the right side of the equation you will know very clearly what they really represent.

(Refer Slide Time: 26:19)


TRANSLATION WITH LOCAL ATTENTION

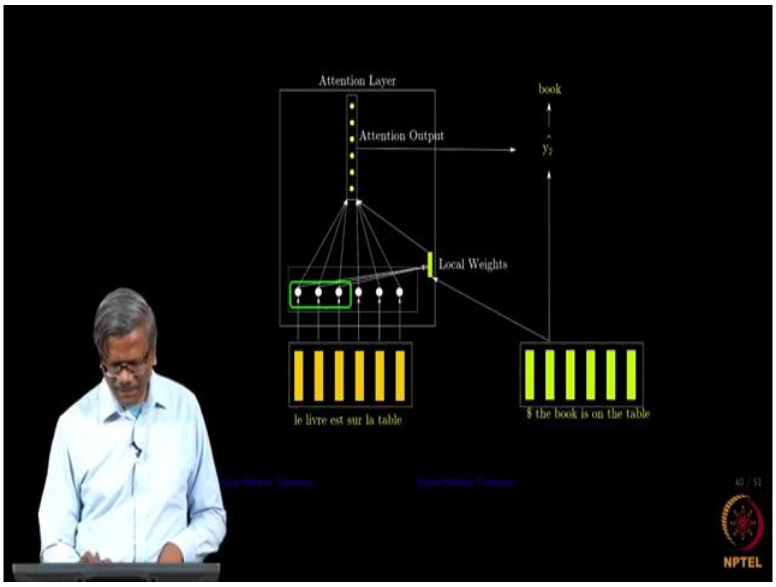
So, now this is what I was talking about in the previous slide, the translation with the local attention. So, rather than peeking at every possible hidden state I want only look at portions of that. Why are we doing this? You remember in the RNN model is this sentence is too long. It is not able to really remember the first part of the sentence right because of the decay in terms of your gradients right. So, even if you use LSTM or GRU you can if I write it.

So, you can see that at this time slot you will have a good representation of that particular state of the input and then slowly and slowly steadily you know decays. So, when your word is here this particular state or will not really know very much about what is there in the x 1 let us say if this is x t. This is the problem with the RNN right. So, we need to overcome this. So, we have several mechanisms to overcome one is the vanishing problems are overcome by GRU and LSTM, but still, we have the decay in terms of or the degraded with respect to knowing something beyond certain states is another problem for this. So, can we somehow overcome this so, that is where the local weights are coming into place?

So, in this case what we are doing is instead of looking at every piece of information that is coming in only look at what I need. So, in this case, again the local weights are computed; the alpha e is computed in the same fashion, alpha is computed in the same fashion and then the context is computed in the same fashion everything remain the same only thing or the modification that we have is only to look at this portion of your h t and not everything else ok.

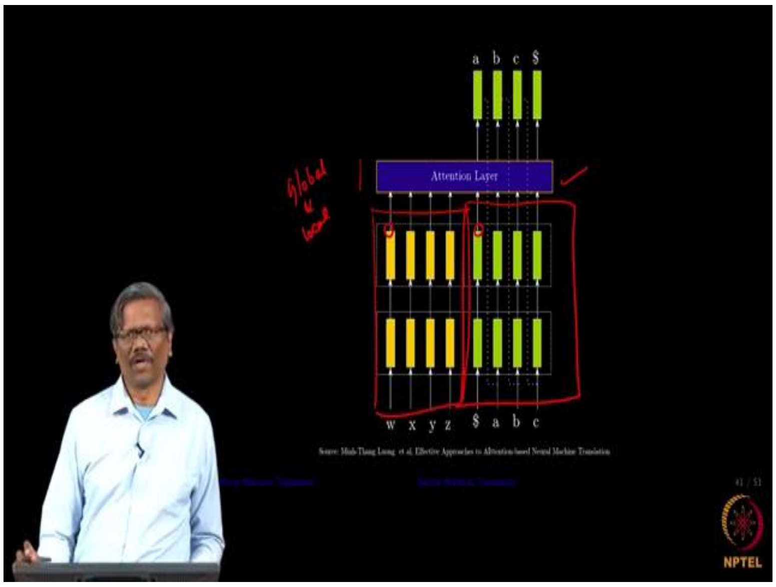
So, by doing this we believe that the contextual information would be very clear and there would not be not much loss in terms of knowing what is the neighborhood for this word here. So, when I say neighborhood for this word here it could be la, le and so on right. So, it should be able to remember the local context very well than the global context that is the idea.

(Refer Slide Time: 29:26)



So, then for the next word if you want to get here we will have three of this, four of this depending on how much context that you want to feed in into your attention vectors ok. So, this is another model that you want to that you may want to know.

(Refer Slide Time: 29:46)


a b c $
Attention Layer
Global & local
w x y z $ a b c
Source: Minh-Thang Luong et al, Effective Approaches to Attention-based Neural Machine Translation
NPTEL

So, at the very high level if you look at this we have included an attention layer; we have the encoder here and then we have the decoder here. So, both of them combined the values from here and here and create the attention whether it is global or local and then that attention layer provides that input to the decoder, and then the output is computed.

So, this is a very standard approach and a simple diagram that affects both global as well as local attention in this in one. So, only here we see how only in this we will introduce the global and local models.