RNN

RNN is used mainly for **Natural Language processing tasks**

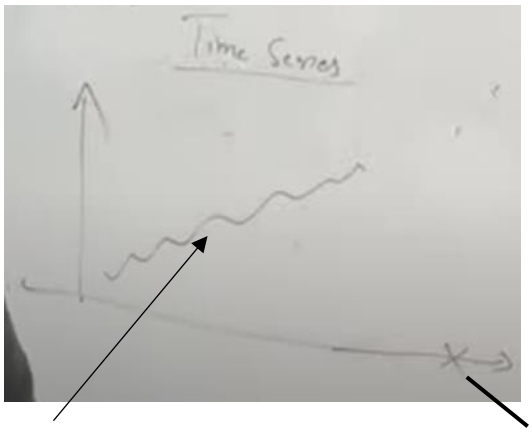
RNN works very well with sequence of data as ur input.

Example where RNN is extensively used – google assistant, amazon alexa

If u ask alexa 'what is the temperature outside'

It will reply u in proper sentence, so replying in a sequence is very important here

Another type of application where its extensively used is Time Series Data



This line is our sales data, at this particular time we need to predict our output.

If u want to predict the output at a particular time using RNN

It will predict based on the timestamps of whole data

Let us consider an example for NLP.

For ex:

- suppose we have a sentence, we have to predict whether this sentence is positive or negative.
- Spam Classifier can be implemented using RNN
- Time Series Data
- Sales Forecasting
- Stock Forecasting and many more things can be done using RNN

Use case –

- **In Gmail when we type a sentence it will autocomplete it**

So Google has this RNN embedded into it,

Where u type in a sentence 'not interested at' – it will be autocompleted with 'this time'

So this saves ur time, it will write the sentence for u

- **Another use case is Translation**, u must have used Google Translate where u can translate a sentence from one to another language.
- **Third use case** is Named Entity Recognition

Where in the X u give neural network a statement and in the Y the neural network will tell u the person name, the company, the time

So these are various use cases where using sequence models or RNN helps.

The fourth use case is sentimental analysis where u have a paragraph and it will tell u the sentiment whether this product review is one star, two star and so on.

Q. Now u would think why can't we use a simple neural network to solve these problems?

Ans: See all these problems they are called **Sequence modelling problems.**

Bcz the sequence is important here.

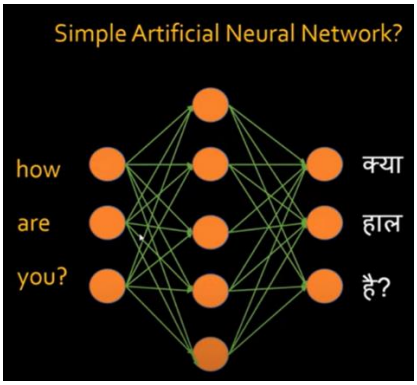
Whenever it comes to human language sequence is very important\

For ex: When u say 'How are you?' vs 'You are how' doesn't make any sense

So the sequence is important here.

And u would think why don't we use simple neural network for that ?

Well, Let's try



For language how about we build this kind of neural network

Here input is the English statement and output is the Hindi Statement

Once I build this what if my sentence size changes. So I might be inputting different sentence size and with a fixed neural network it's not possible

Bcz u have to decide how many neurons are there in the input layer and output layer as well.

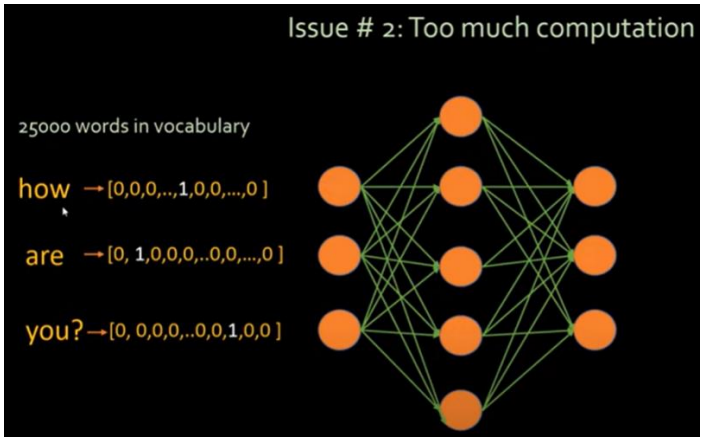
- So with language translation number of neurons bcms a problem

Like what do u decide as a size of neurons, one might say that we can decide a huge size lets say 100 neurons

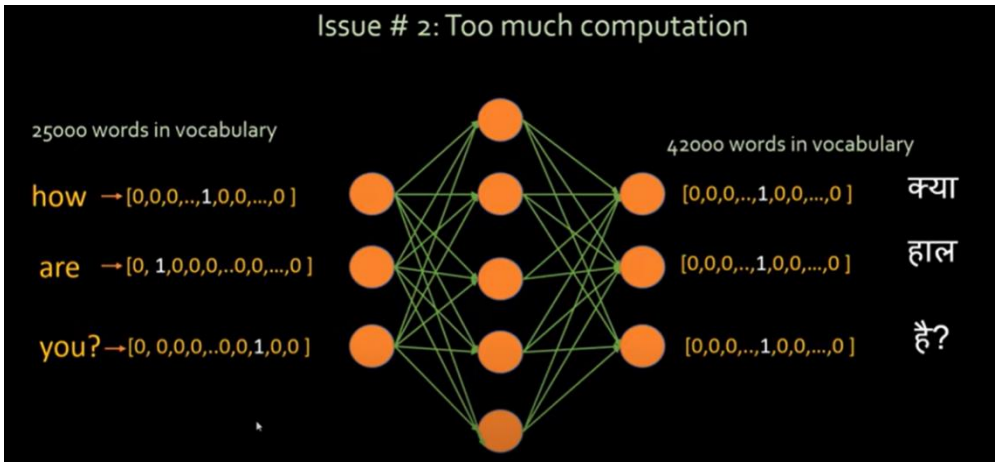
And remaining? If u say 'did you eat biryani'. It will occupy four neurons

Remaining 96 I will say 0 or blank statement

That might work but still it's not ideal



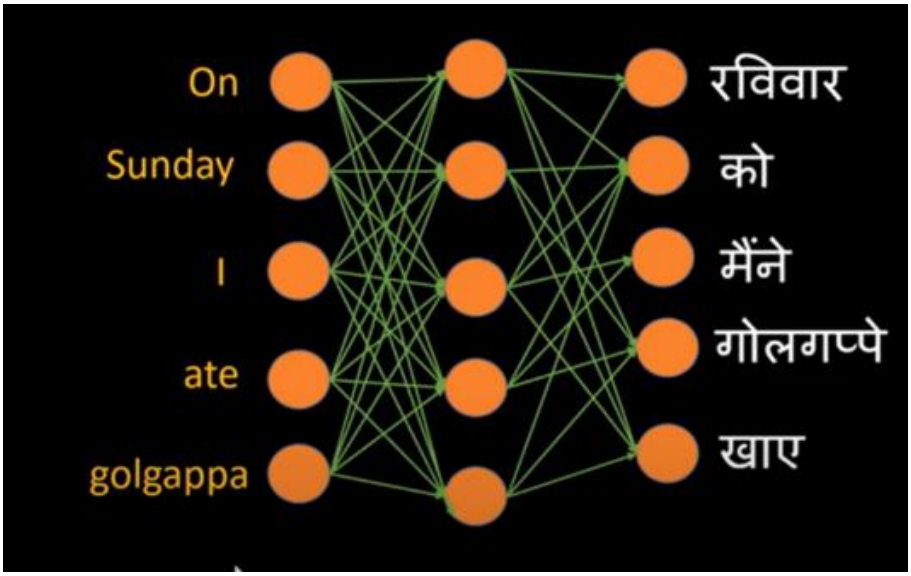
- The 2nd issue is too much computation. U all know neural network work on numbers not on string. So u have to convert ur word into a vector
  So one of the way of converting that into a vector is
  Let's say there are 25,000 words in ur vocabulary & u will do one hot encoding
  Let's say 'how' is in 46th position
  'are' is at 2nd position
  'you' is at 17000th position
  So at that position u put 1, remaining position u put 0 and that's called one hot-encoding.



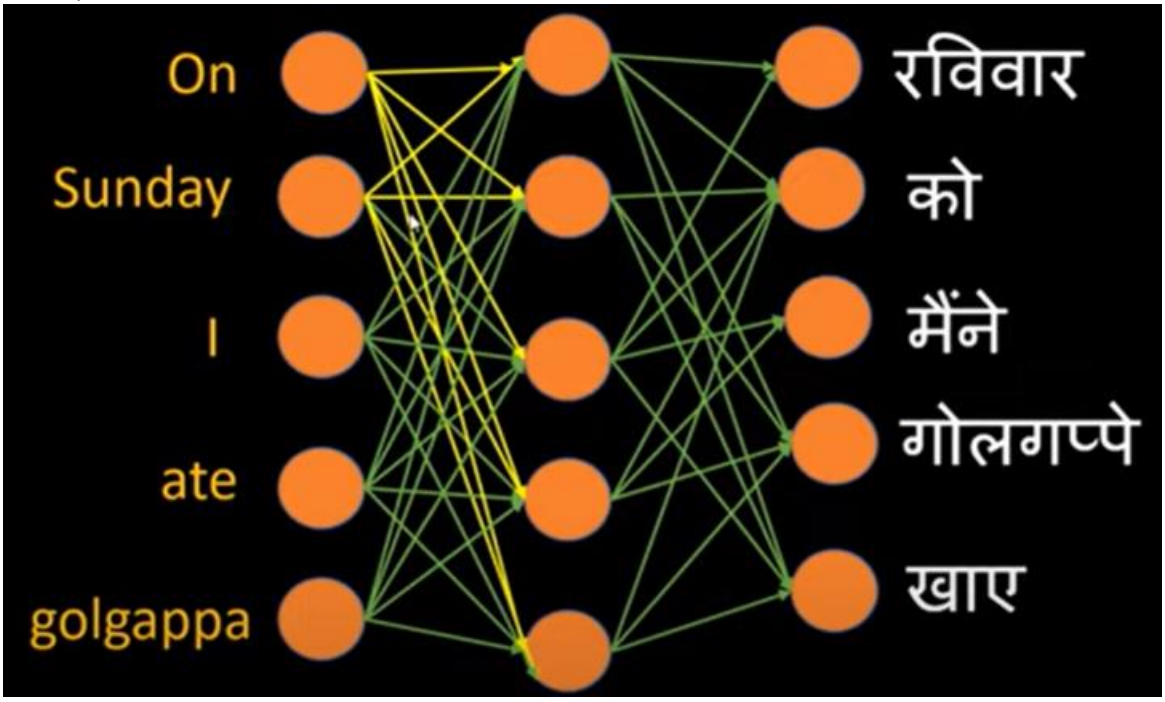
You have to do similar thing for output as well, but u realize this will increase too much computation
Each of the word when u convert into a vector u know how many neurons it will need in input layer itself. It's humongous

- **The third issue is**

Sometimes when u translate language,
for ex: u say two different english statements
u might have a single hindi statements

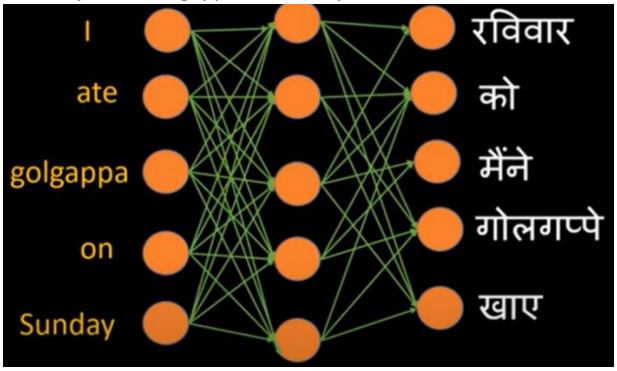


When I say 'On Sunday I ate golgappa
Let's say we trained this network based on this statement

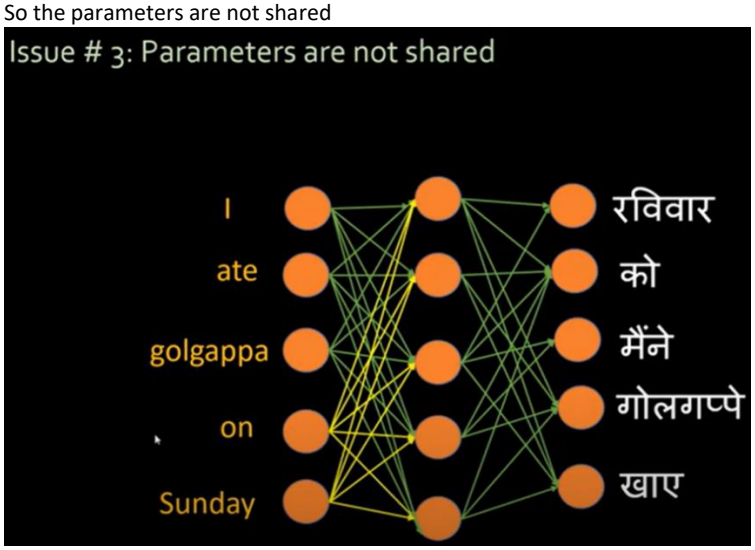


And for 'On Sunday' let's say it will adjust the weights of all these edges which I have highlighted in yellow colour

Same statement I can say differently
I can say 'I ate Golgappa on Sunday'.

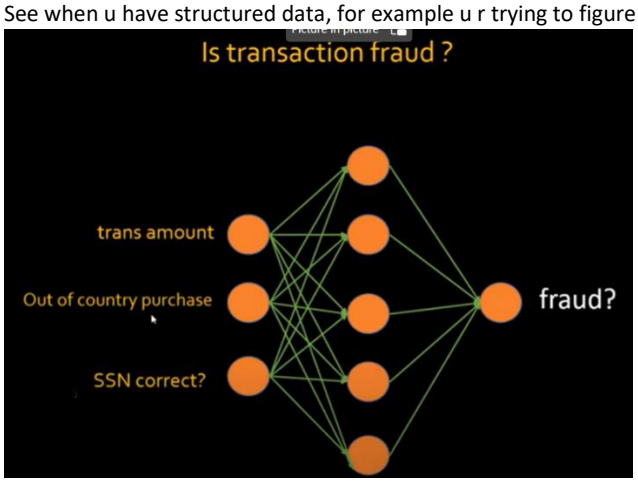


So the meaning of ‘On Sunday’ is same but here neural network has to learn different set of edges u see all these edges are in yellow colour
So the parameters are not shared

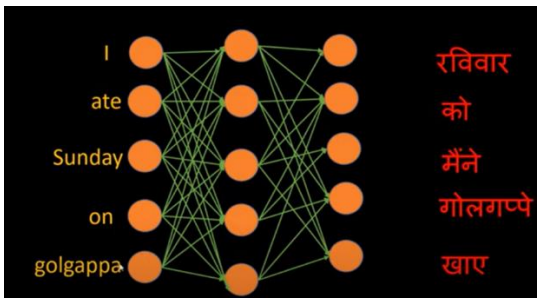


Also the most important part in all these discussion is the sequence

See when u have structured data, for example u r trying to figure out if the transaction is fraud or not



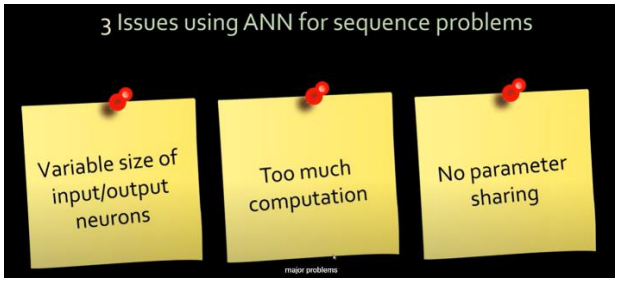
And let’s say ur features are transaction amount, whether the transaction was made out of country or whether the SSN that customer provided is correct or not
Now here if u change the order of these features let’s say we put SSN correct in place of trans amount
Its not gonna affect anything
Bcz here the sequence in which u supply the input doesn’t matter.



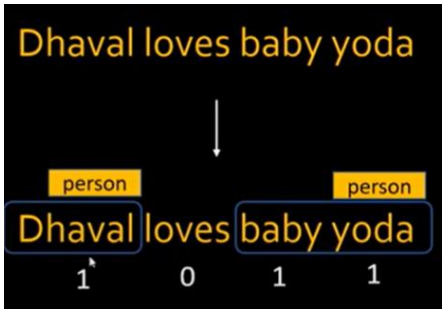
Whereas if u have English to Hindi translation and instead of saying ‘I ate Golgappa on sunday’ if I say ‘I ate Sunday on Golgappa’
The meaning becomes totally different
So now u cannot say that the hindi translation is ‘ravivar ko mene golgappe khaye’
Bcz it bcms invalid.

So sequence is very much important that's why artificial neural network doesn't work in this case.

To summarise…

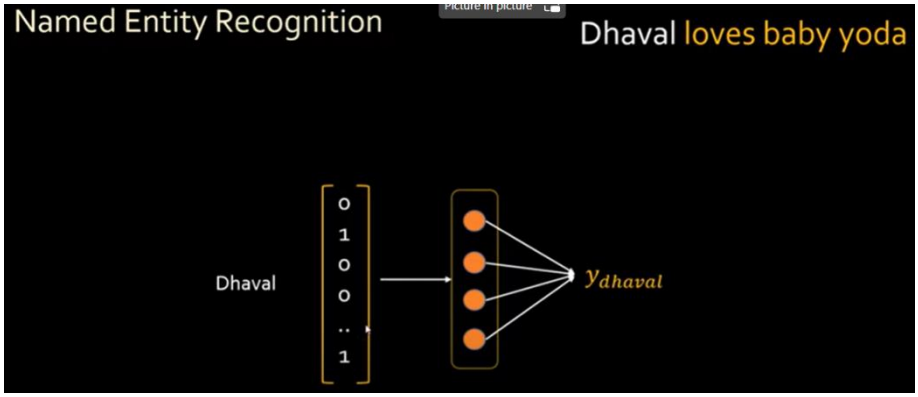


Let's once again see about Named Entity Recognition

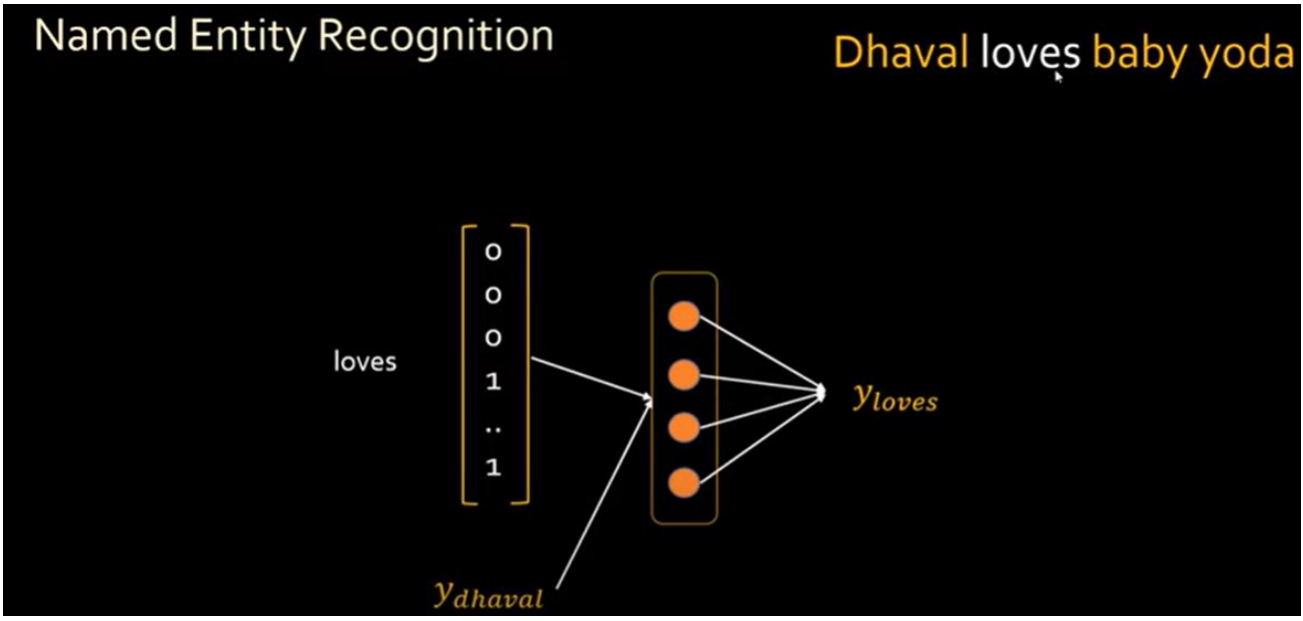


In this statement Dhaval and baby yoda are person names
The whole purpose of Named Entity Recognition is to find out the entity you know like 'Dhaval' as an entity as a person, baby yoda as an entity as a person
So that's the whole goal of Named Entity Recognition
Now u can represent this as one's and zero's
So if the word is person's name you would mark it as 1
And if it is not person's name u would mark it as 0.

So let's see how Recurrent Neural Network works here..

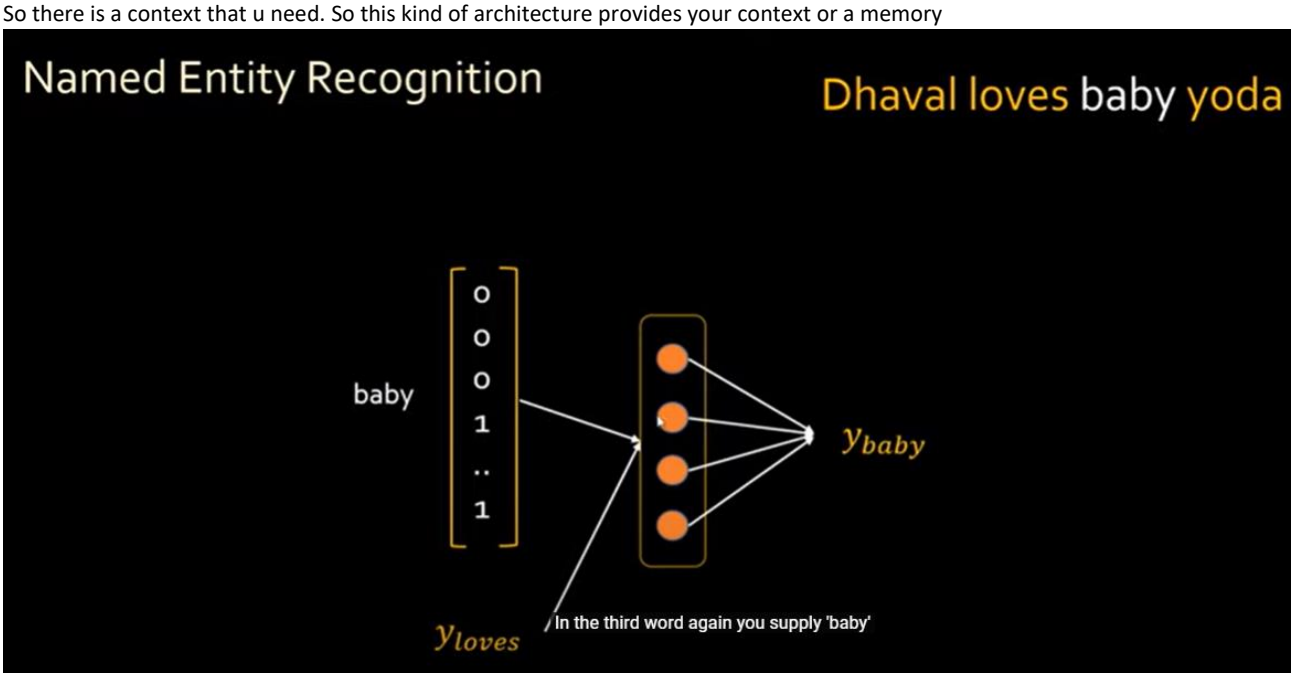


So first of all u have to convert dhaval into some vector
U can convert it by hot-encoding and there are other ways of vectorizing a word
Then u have a layer of neurons, so these are all individual neurons
Let's say that is one layer, it's a hidden layer
U supply that and u get one output
So each neuron as u know has a sigma function , activation function
So now processing the statement Dhaval loves Baby Yoda
Now we will process it word by word
So I supply 'Dhaval' get the output and then I go back again

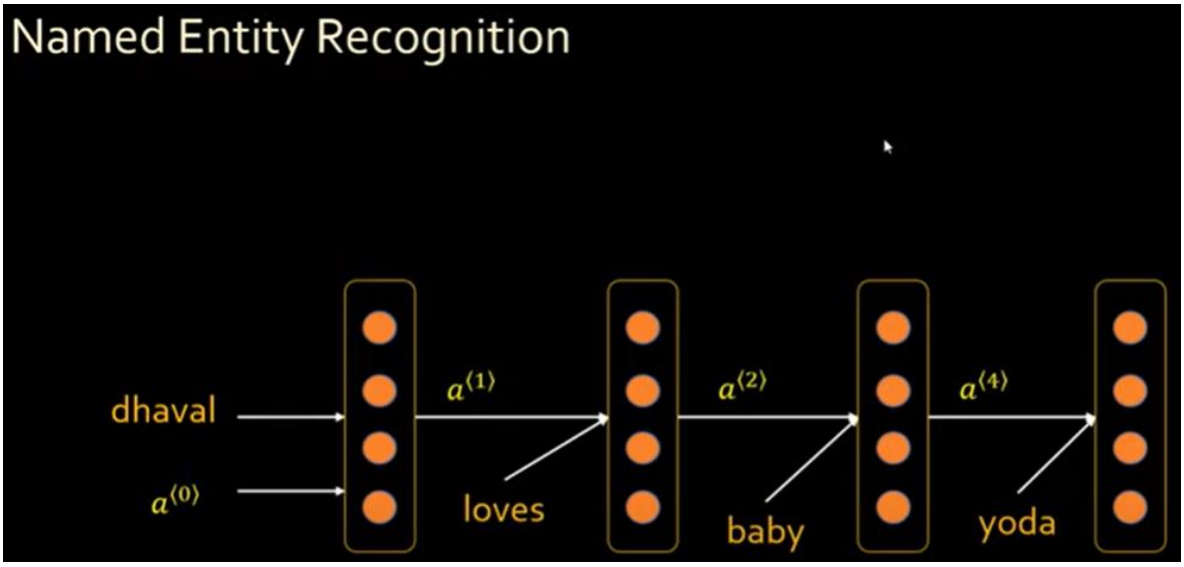


Now I supply loves convert it into vector and the previous output which I got which was ydhaval I now supply that as a input to the layer
So the layer will get 2 inputs
So the input of the layer is not only the next word but the previous output bcz the language makes sense
Language needs to carry the context
If have just a word 'loves' and I don't have 'Dhaval' in front of it - it might mean a different thing

So there is a context that u need. So this kind of architecture provides your context or a memory



Again u supply the third word 'baby' to the network
So our network has only one layer
So our input layer, hidden layer and output layer is just one and it has bunch of neurons
In that we are repeatedly processing word one by one and u keep on doing this
So benefit of doing this is when I am processing baby when I get loves that loves carrries the previous state or previous memory of 'dhaval loves' the whole statement



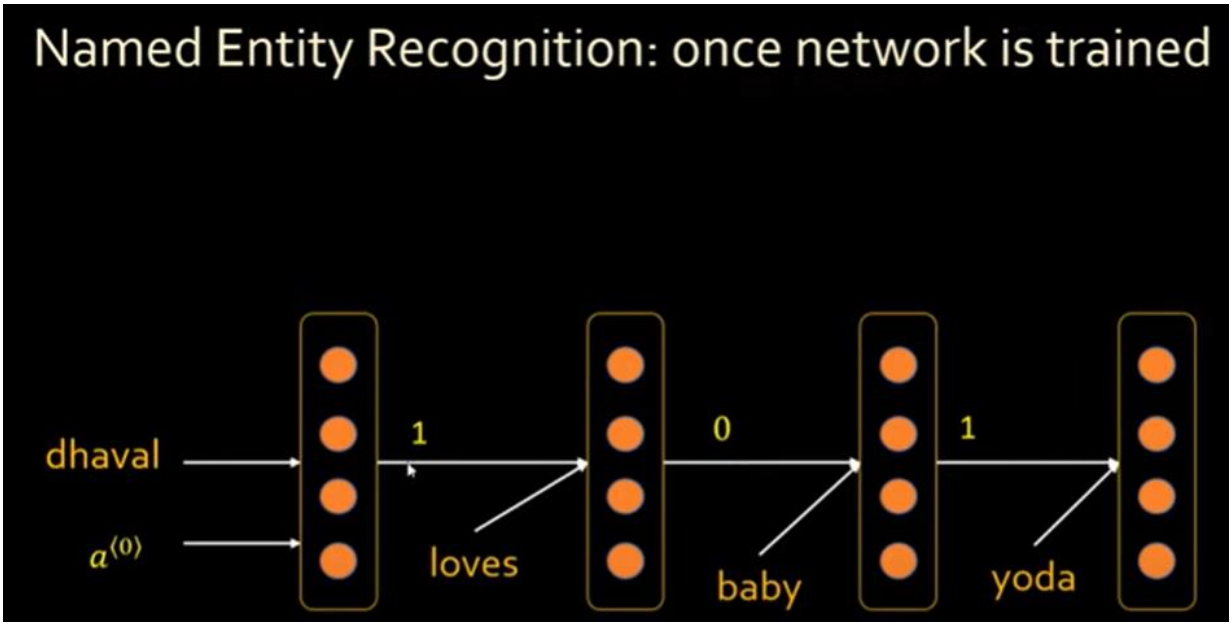
Now representing the same thing in a different way

**** Remember that these are not four different hidden layers**
This is a timetravel okay, so actual hidden layer is only one

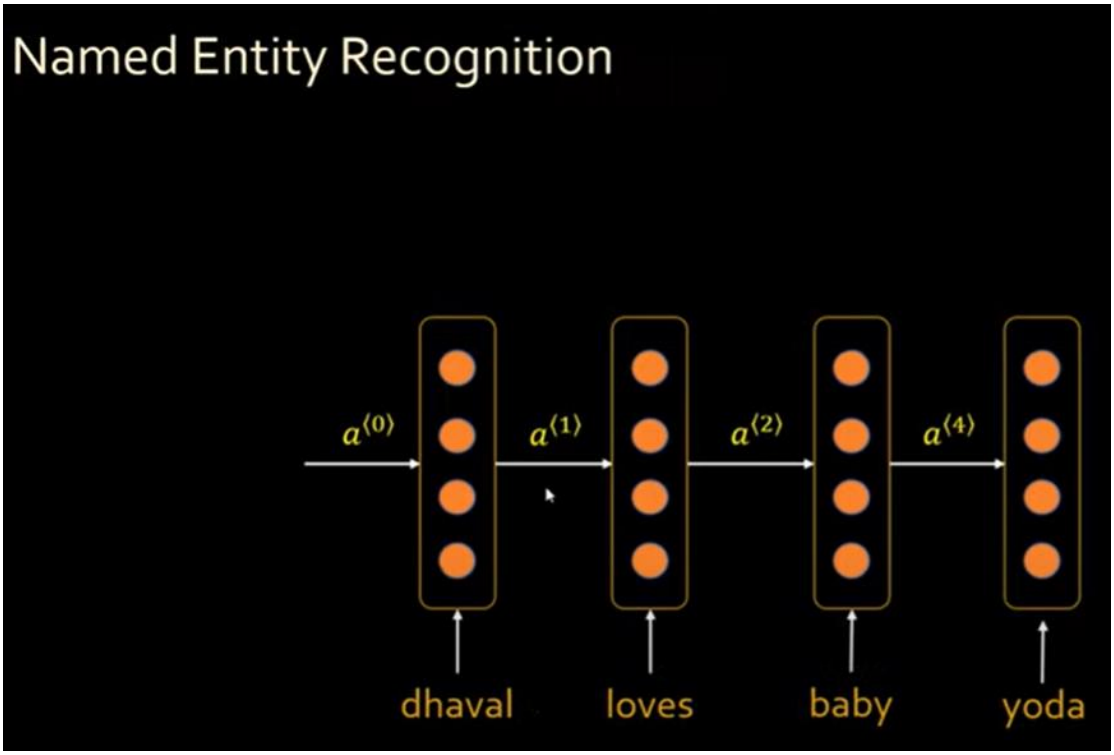
So first when we supplied the word 'dhaval' we got the ouput as a(1) and a(1) is nothing but the activation function which I am denoting with a(1).
And u need some previous activation function a(0) as well
Let's say a(0) is a vector of all zeroes
then u supply 2nd word 'loves' and use the previous output which was ydhaval and a(1) both are same
and then u get another output a(2), that u supply along with the third word 'baby' to the same network

So this 4 neurons it's the same single layer
I am just showing the status of it at different times

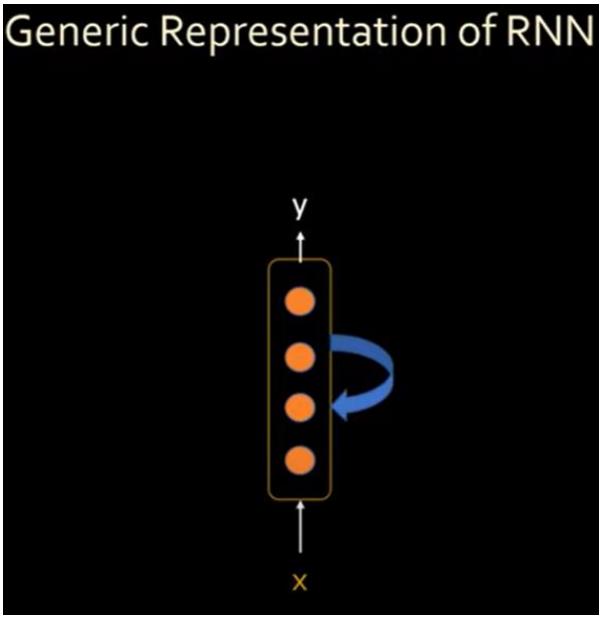
And once the network is trained ofcourse it will output like 'Dhaval' is 1 bcz it's a person name
And Dhaval loves is 0 bcz loves is not named entity
And Dhaval loves baby is 1 bcz baby is a person i.e entity
And so on…



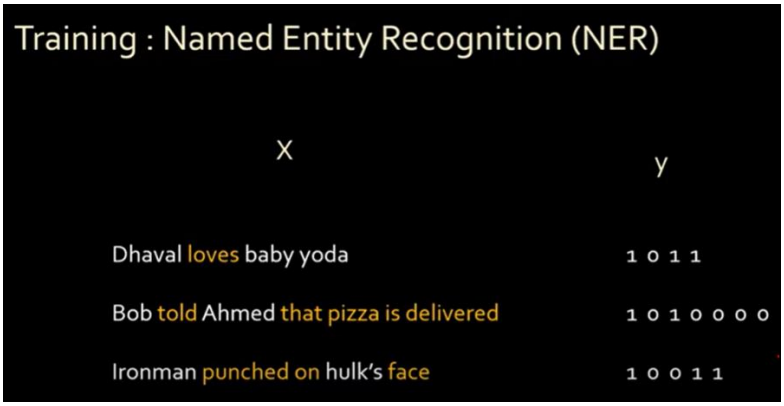
So u get ur Named Entity Recognition output individually here



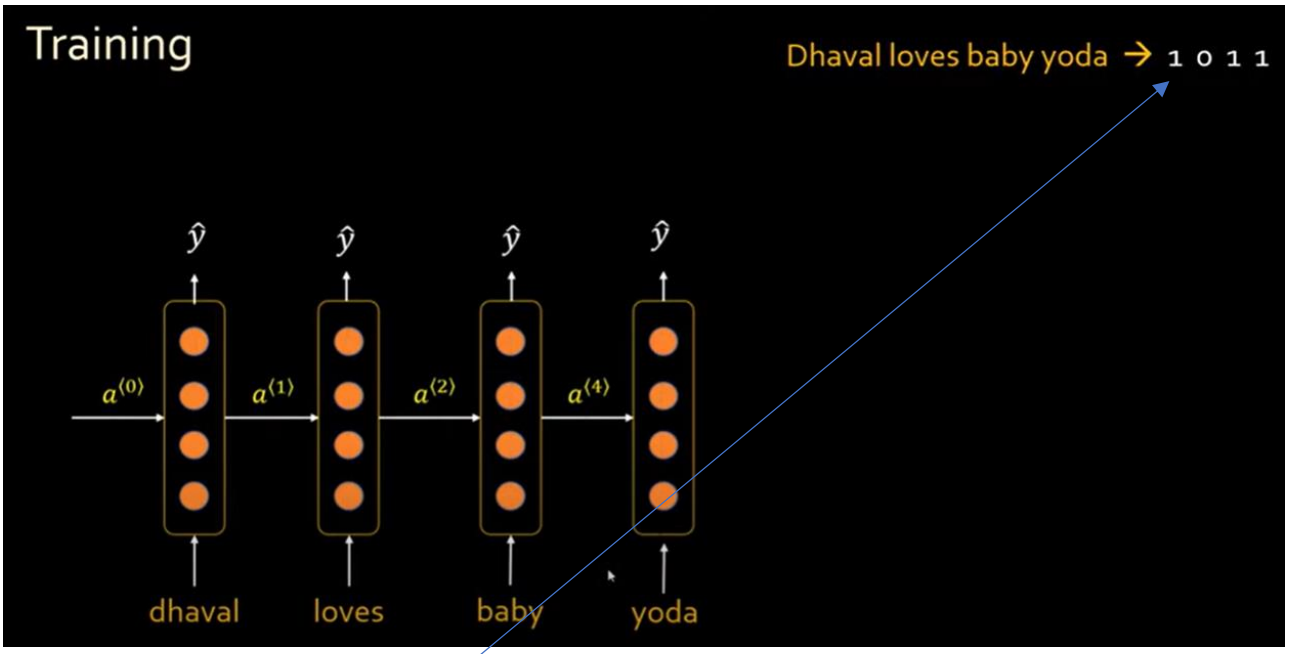
Above fig is another way of representing the network



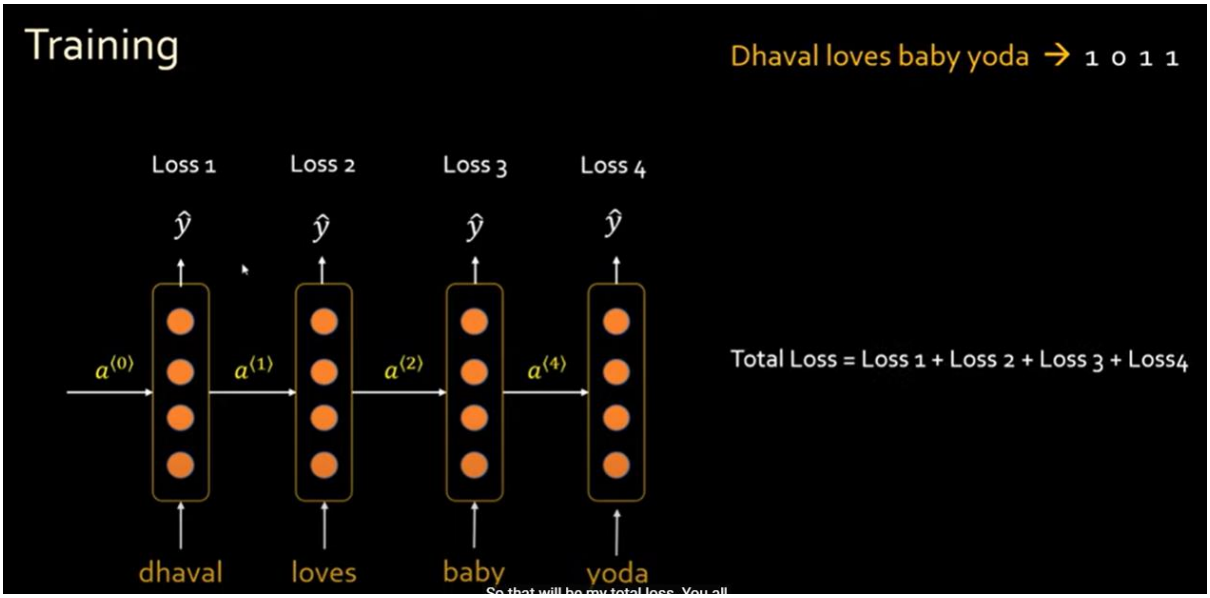
This is the real representation, u have only one layer and u r kind almost in a loop
U r supplying the output of previous word as an input to the 2nd word



So now let's talk about training
So again the problem we are talking about is Named Entity Recognition where these are our training samples
X is a statement
Y is whether given word is person name or not

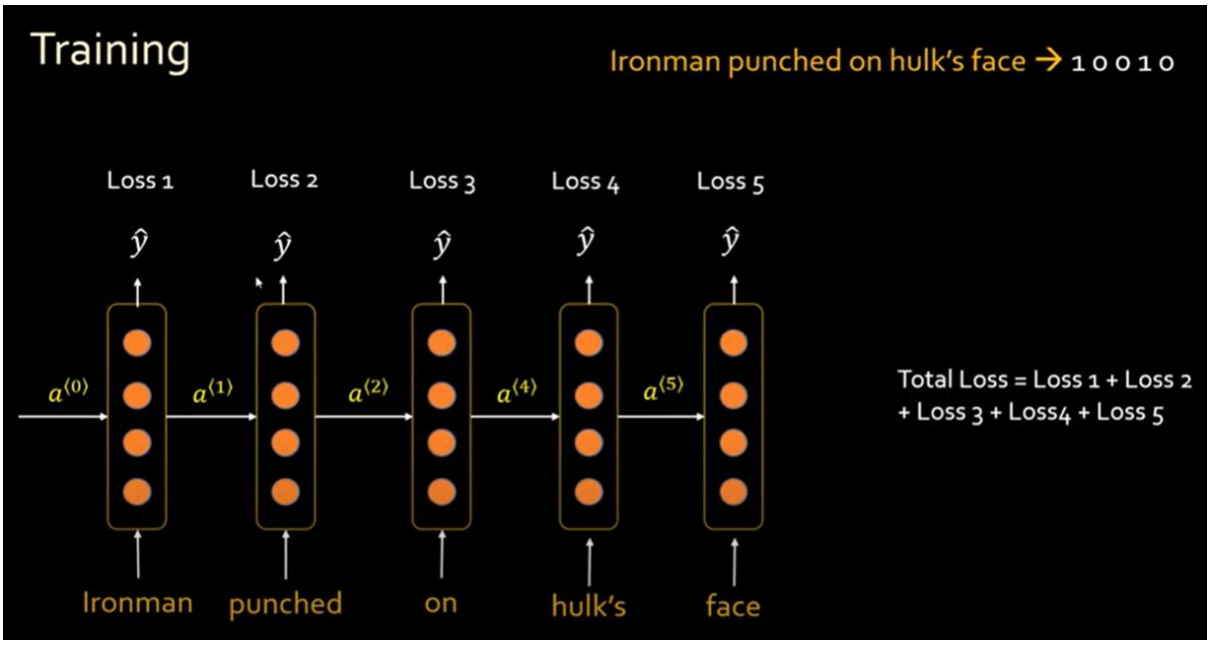


We are processing first training sample
First we will initialise my neural network weights(a0,a1,a2,a4) with some random values.
Then I supply each word (Dhaval,loves,baby,yoda)
Then we calculate y^ which is predicted y
Then, I compare with the real y 1 0 1 1



And then we find out the loss and then sum the loss, that will be my total loss
We compute the loss and then we back propagate the loss and we adjust the weights

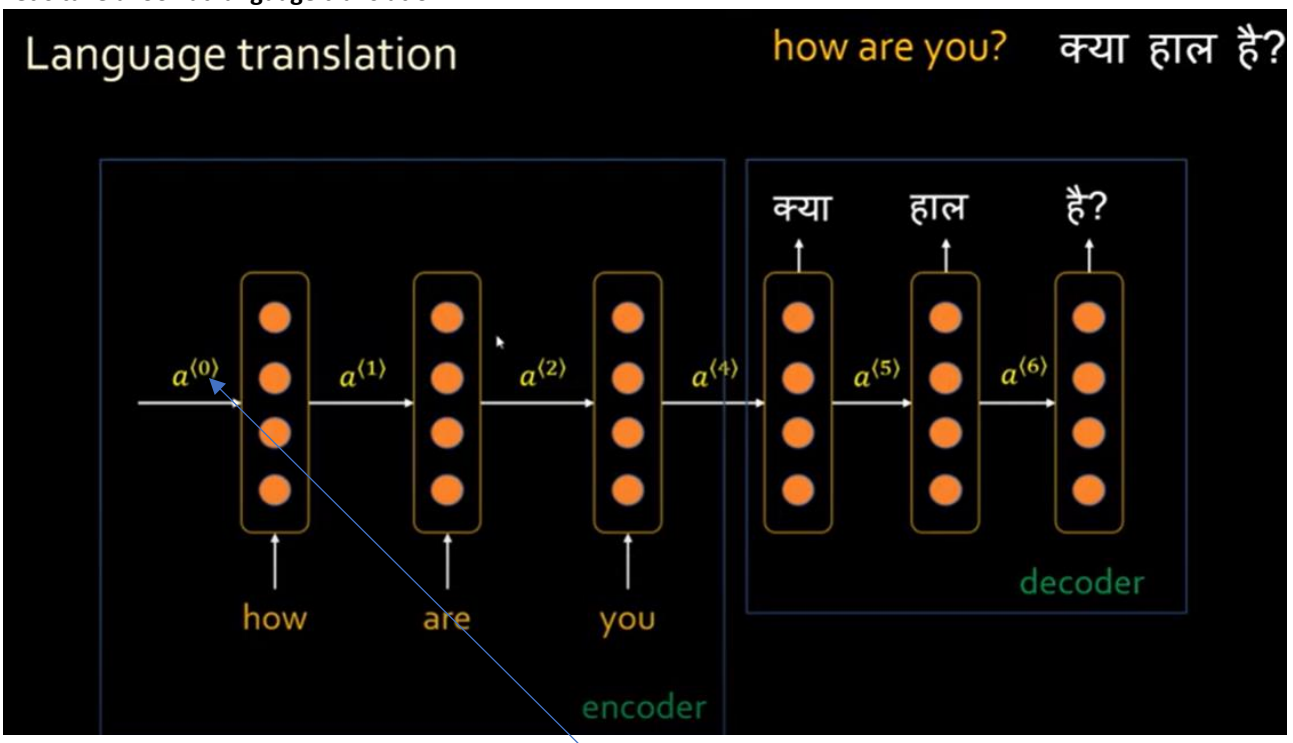
Let's take the 2nd statement



Again we calculate losses and then we find total loss
And then I do gradient descent to reduce the loss

We keep on doing this for all training samples
Let's say we have 100 training samples
Passing all the 100 training samples through this network would be 1 epoch
We might perform 20 epoch, after 20 epoch my loss would be minimum
At that point I can say my neural network is trained

Let's take a look at language translation



In language translation what happens is u supply first word to ur network then u get the output
Then again u supply 2nd word and the output from previous step as an input to same network
& ofcourse when u supply 1st word u have to pass in some activation values (lets say a vector of all 0's)

And then u supply 3rd and 4th word and so on

And when u r done with all the words

That's when the network starts to translate bcz u can't translate one word by one bcz after the statement I can push may be one more word and that will just totally change my translation

That's why for language translation u have to supply all the words and only then the network can translate for u.

So network will translate like this and the 1st part is called the encoder and 2nd part is called decoder.

Now this layer doesn't have to be just single layer. It can be deep RNN as well where the actual network might have multiple hidden layers