Our basic RNN model is like Ghajni

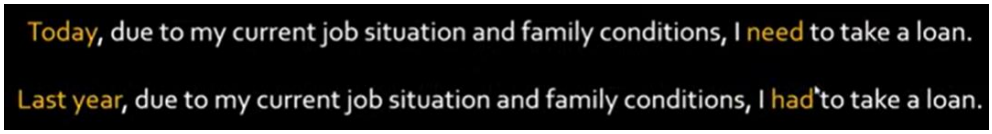
It suffers from short-term memory problem.

LSTM is a special version of RNN

LSTM solves the short-term memory problem.

We have a NLP task for sentence completion

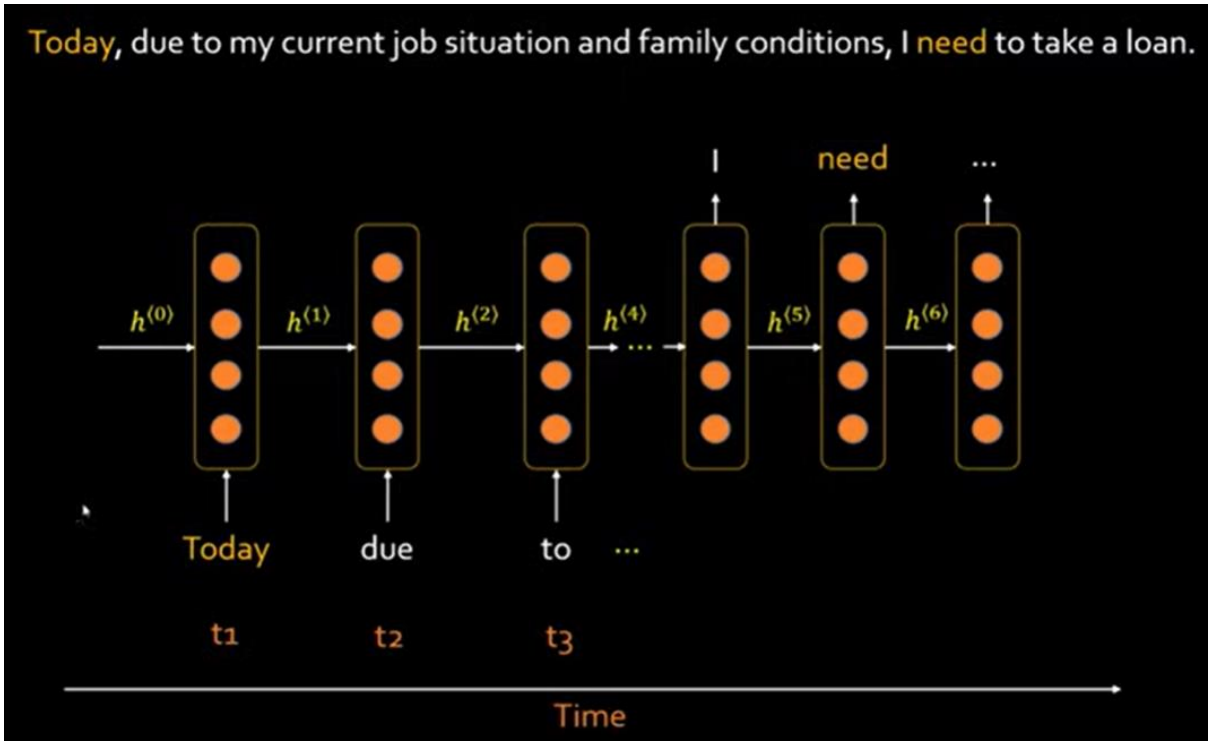
Here in both the sentences based on what word appeared in the beginning ur autocomplete sentence might be different.



For the first one I would say I need to take a loan

Whereas for the 2nd one , I would say I had to take a loan

And this decision between need and has was made based on what appeared at the very beginning.



To predict this word 'need' we need to know about the word 'today'.

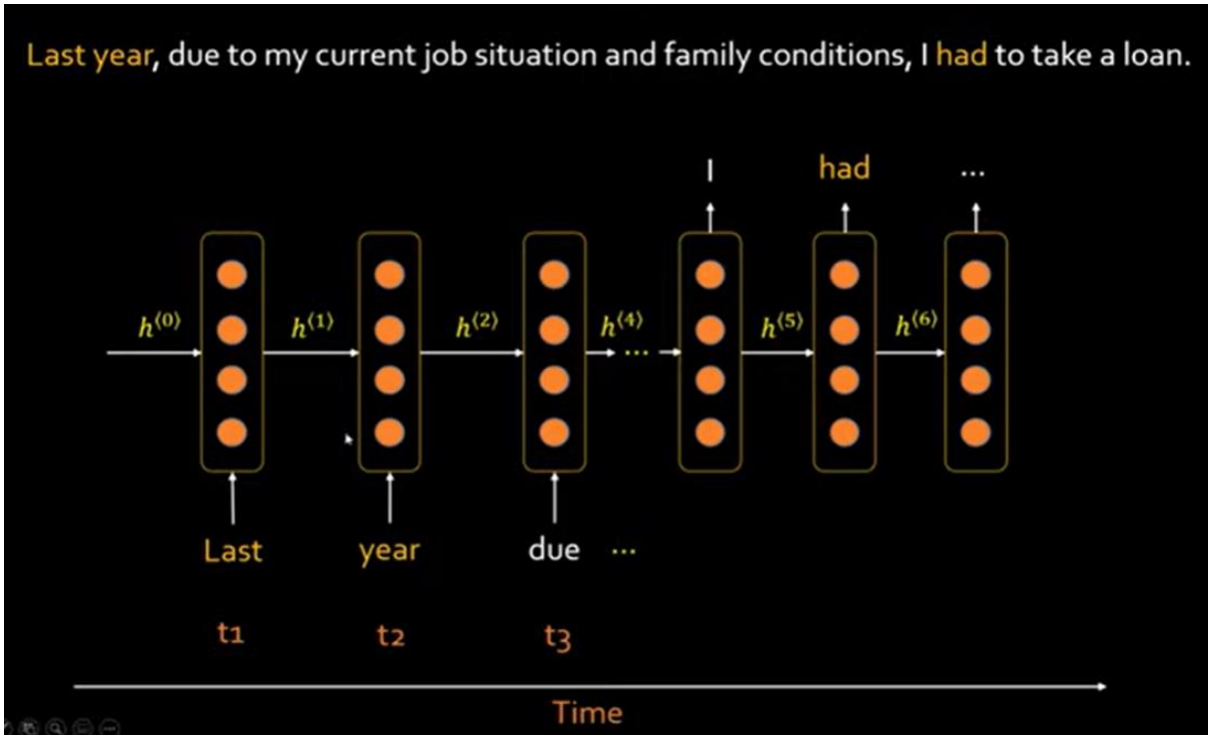
Which appeared at the very beginning of the sentence

And bcz of vanishing gradient problem the traditional RNN's have short term memory,

So they don't remember what appeared in the beginning of the sentence

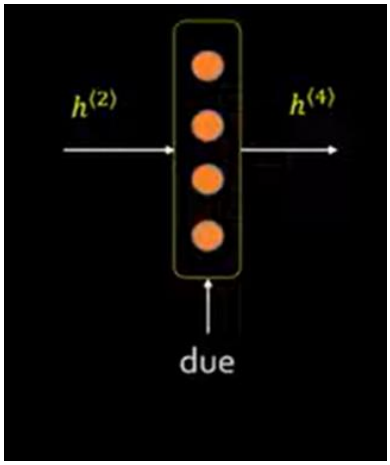
They have very short term memory of very few words which are nearby

Hence to autocomplete this kind of sentence, RNN won't be able to a good job.



Now lets look at network layer in detail

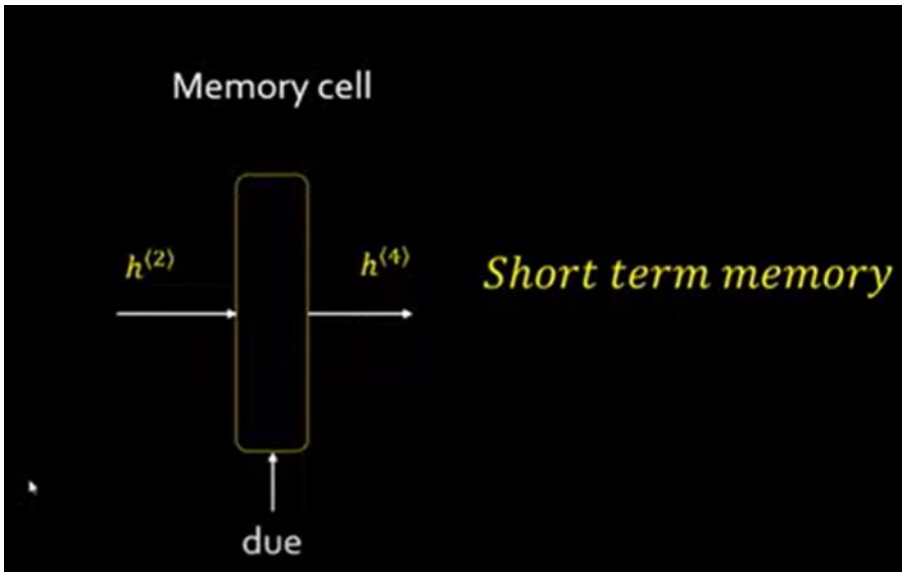
Just expanding the network layer



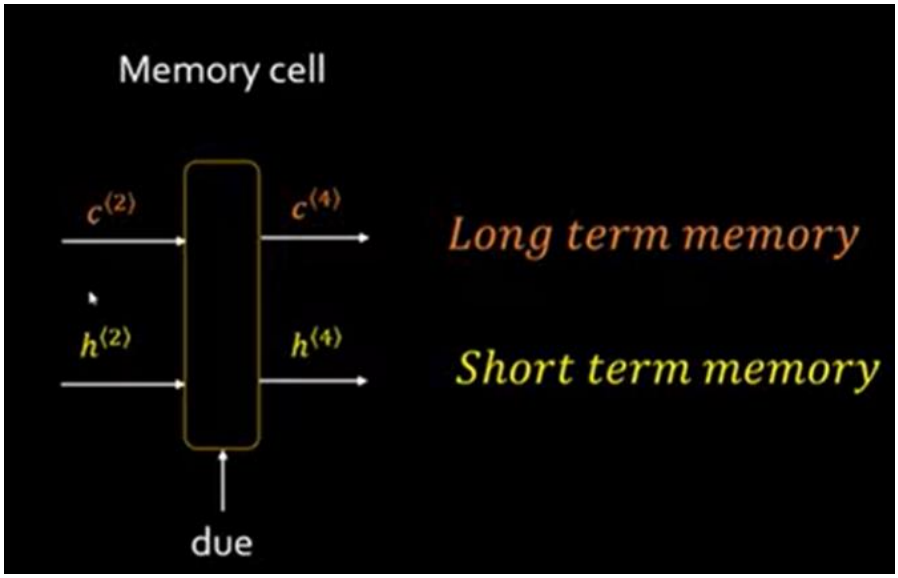
So there are set of neurons in the layer and this hidden state is northing but short term memory

Just remove this neurons and make it simple

This square box is called the memory cell bcz this hidden state is actually containing the short term memory.



Now if u want to remember long-term memory u need to introduce another state called long term memory

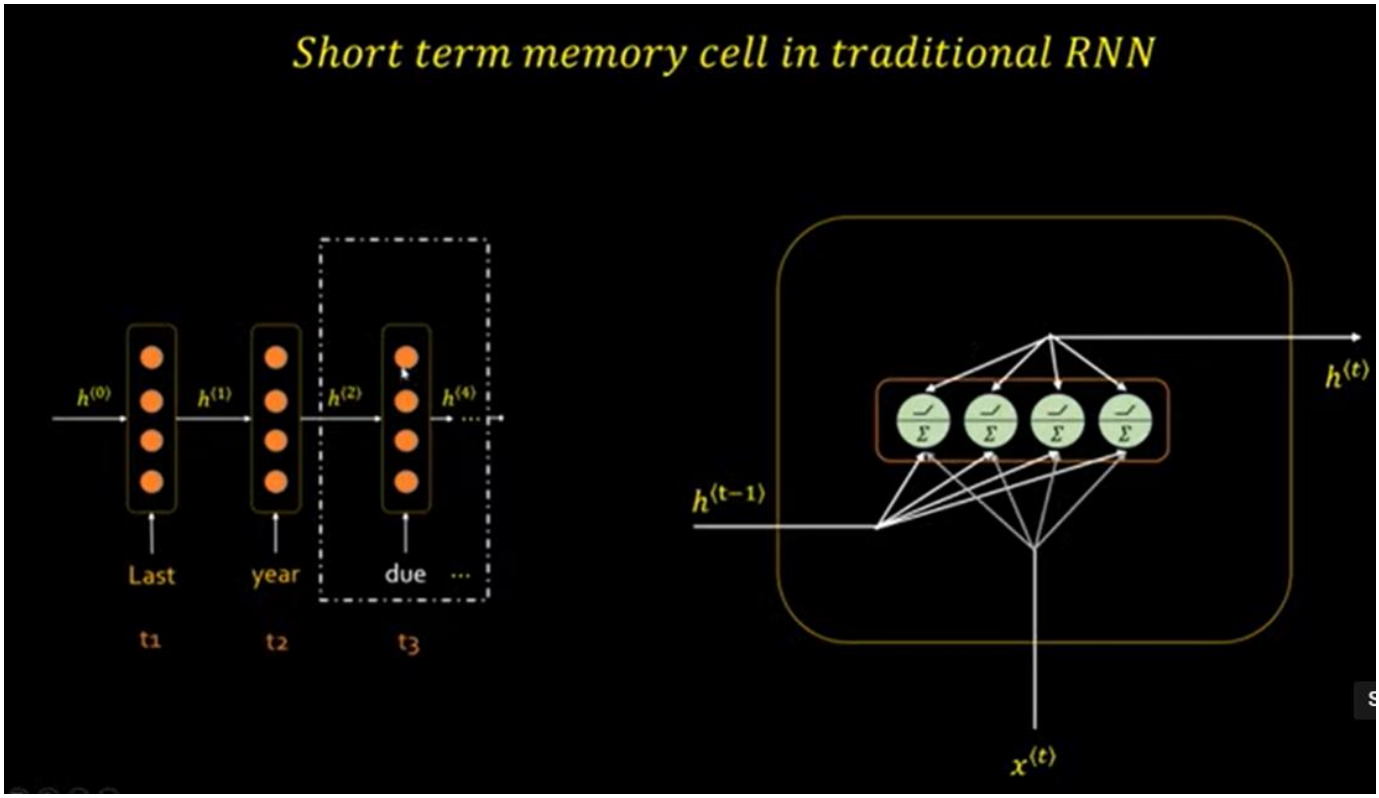


So this state is called c

So there are two states now

Hidden state which is short term memory

And there is a cell state which is a long term memory

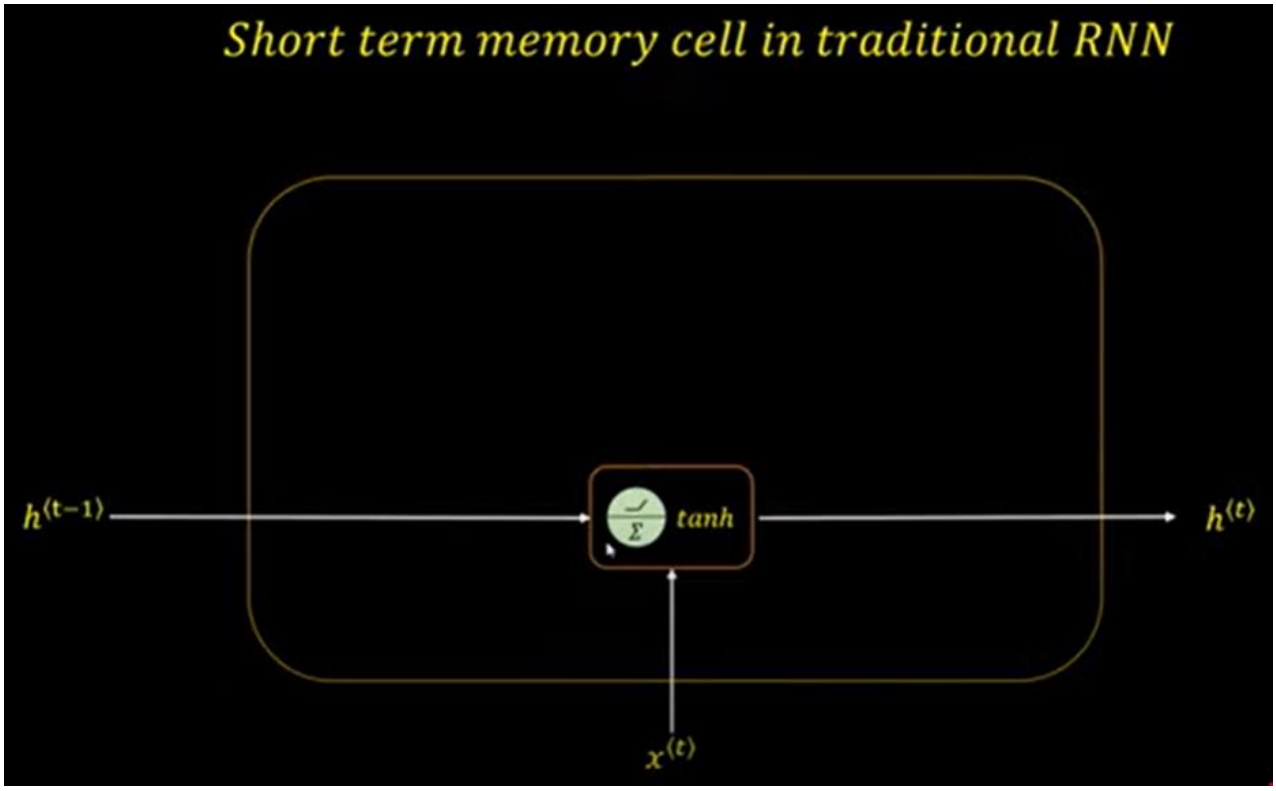


When u have a word u will first convert into a vector, vector is nothing but a list of numbers

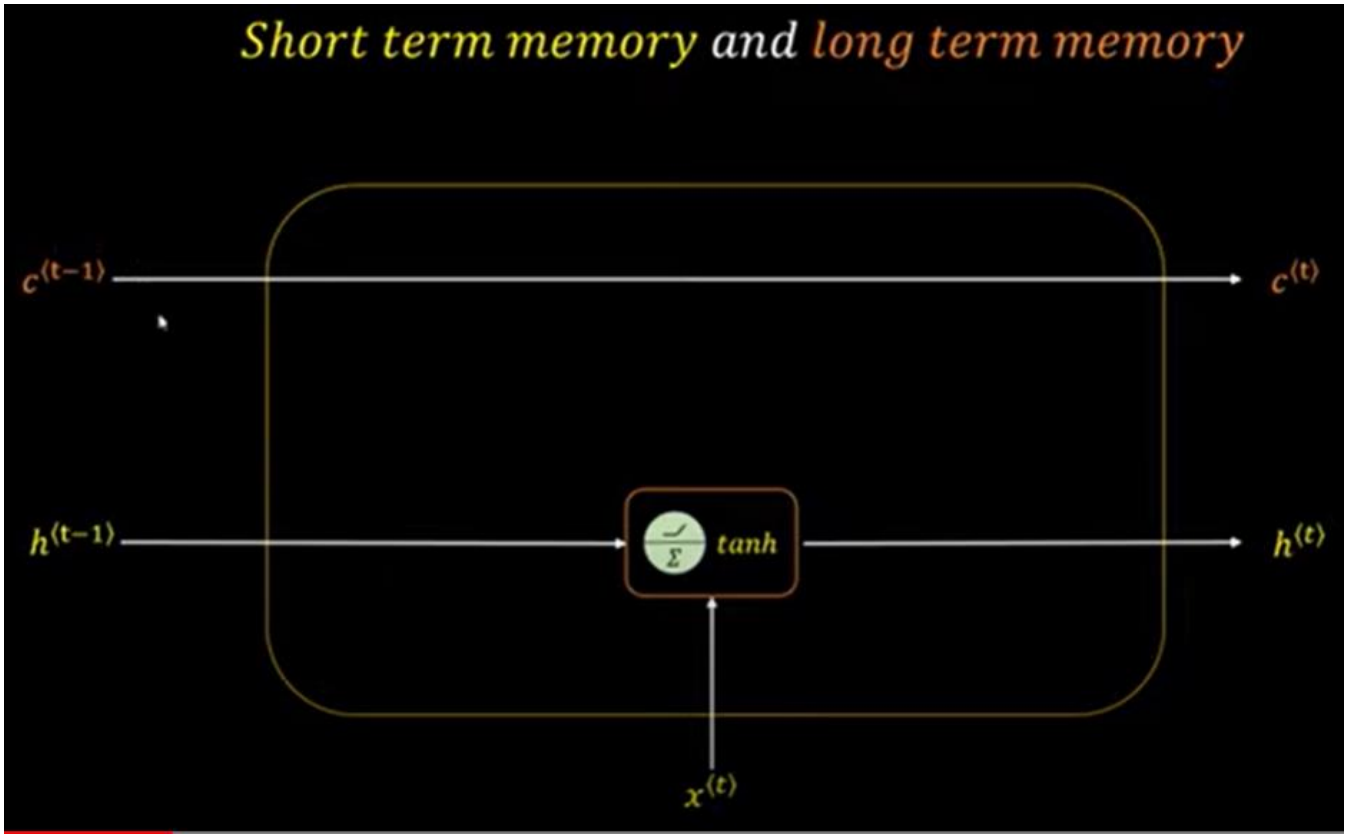
& ur input state (x^t) and hidden state(h^t-1) both will be vectors

Using both these vectors u will do weighted sum multiplication and then u apply activation function which is tanh in the case of RNN

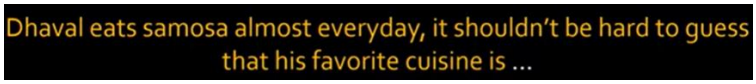
And then u get a new hidden state h(t)



So here is a simplistic example



In LSTM we are going to introduce a new cell state for a long term memory



We have one more sentence to autocomplete

Obviously we will put Indian in the blank

Bcz Samosa is an Indian cuisine so we will say his favourite cuisine is Indian

When u r processing this sentence which words told you that this will be an Indian cuisine

Well it is this word 'Samosa'

If we did not had samosa here in the sentence, if we only had 'eats almost everyday'

So based on these words u cant guess that it is an Indian cuisine right?

There is some key words.

If we are doing a movie review, we look for key words like excellent or terrible movie, or amazing

So we look for specific words and the remaining words we can actually ignore

Lets see how traditional RNN would behave for this sentence

So 'Aamir khan of Gajni having short term memory'

When u feed all these words it can remember let' say last two words

In reality RNN can remember more words but we are just taking a simple example

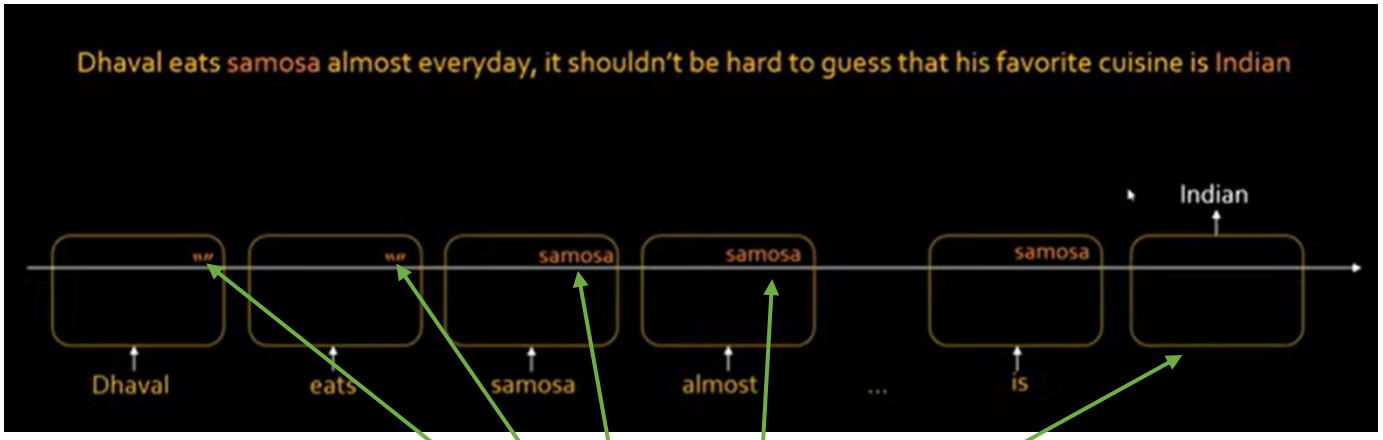
So lets say we have short term memory which remembers only two words

When u r at this word , it remember 'almost' and the 'samosa' last two words

When u r at here, it will remember 'is' and 'cuisine'.

So at this point , it doesn't have knowledge of samosa

So then for a traditional RNN it is hard to make a guess that the cuisine is 'Indian'



What if we build this long term memory along with short term memory in such a way that we store the meaningful words or the key words into this long-term memory

So when I feed Dhaval or eats it will not store, it's a blank string

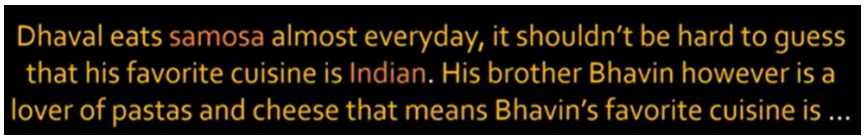
It will not store these words in long term memory

But when u find things like samosa it will store in long term memory

After that 'word' almost , 'almost' is not important so we just store samosa here

Now when I have to make prediction on cuisine, we have that memory that we are talking about samosa and hence it has to be Indian

Lets look at little more complicated example



Here while Dhaval loves Indian cuisine , bhavin loves which cuisine any guesses ?

If u read the whole sentence carefully u will figure it as an Italian Cuisine

And u made that decision based on the two key words pastas and cheese

So here we read the sentence and we are keeping some keywords in memory and throwing everything out words like 'and' , 'that', ',means' these are not important

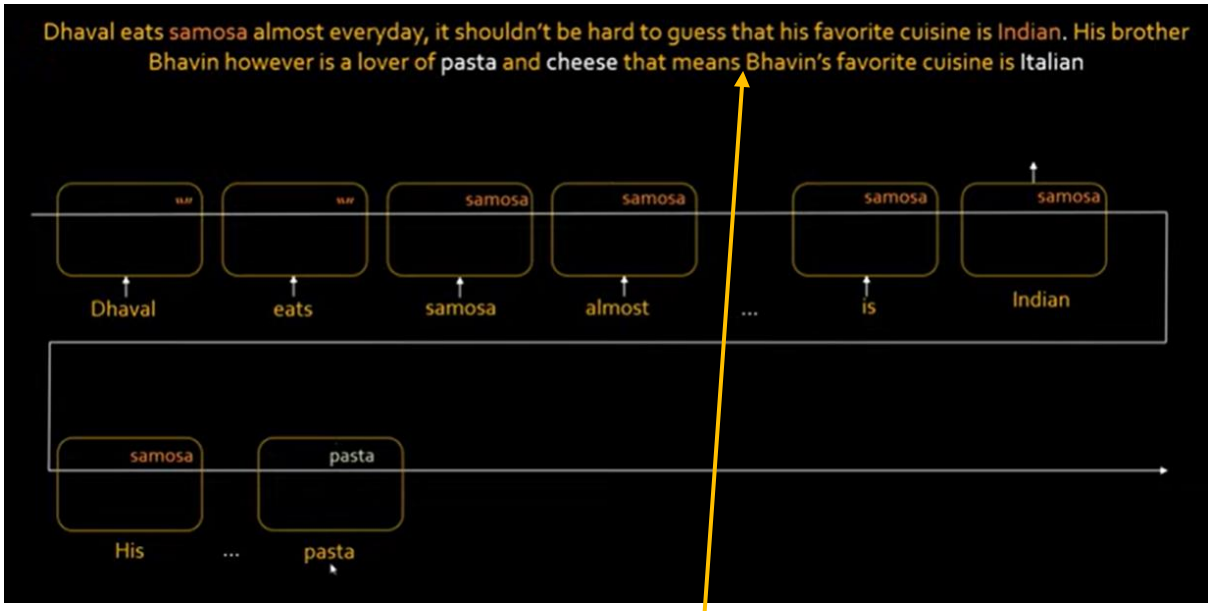
For us important keywords are pastas and cheese

Based on that u make a decision that it is a Italian cuisine

So going back to our RNN with long term memory when u encounter let's say samosa

U will store samosa in ur long-term memory but u will keep on storing samosa until u encounter pasta

So now the moment u encounter pasta you need to forget the previous long-term memory which is samosa



So here we threw samosa out and we will have new memory which is pasta and u keep on preserving this until u hit cheese

So when u hit cheese u need to add that so now u can't ignore pasta , u need to add cheese on top of pasta

And then in the end when u r about to be asked the answer for auto-complete u will say 'Italian'. Bcz u have the memory of pasta and cheese .

Q. Now a question arises how can we make the decision

Lets say when cheese comes u don't discard pasta but when pasta comes u discard samosa.

Ans: well all of this happens during the training process

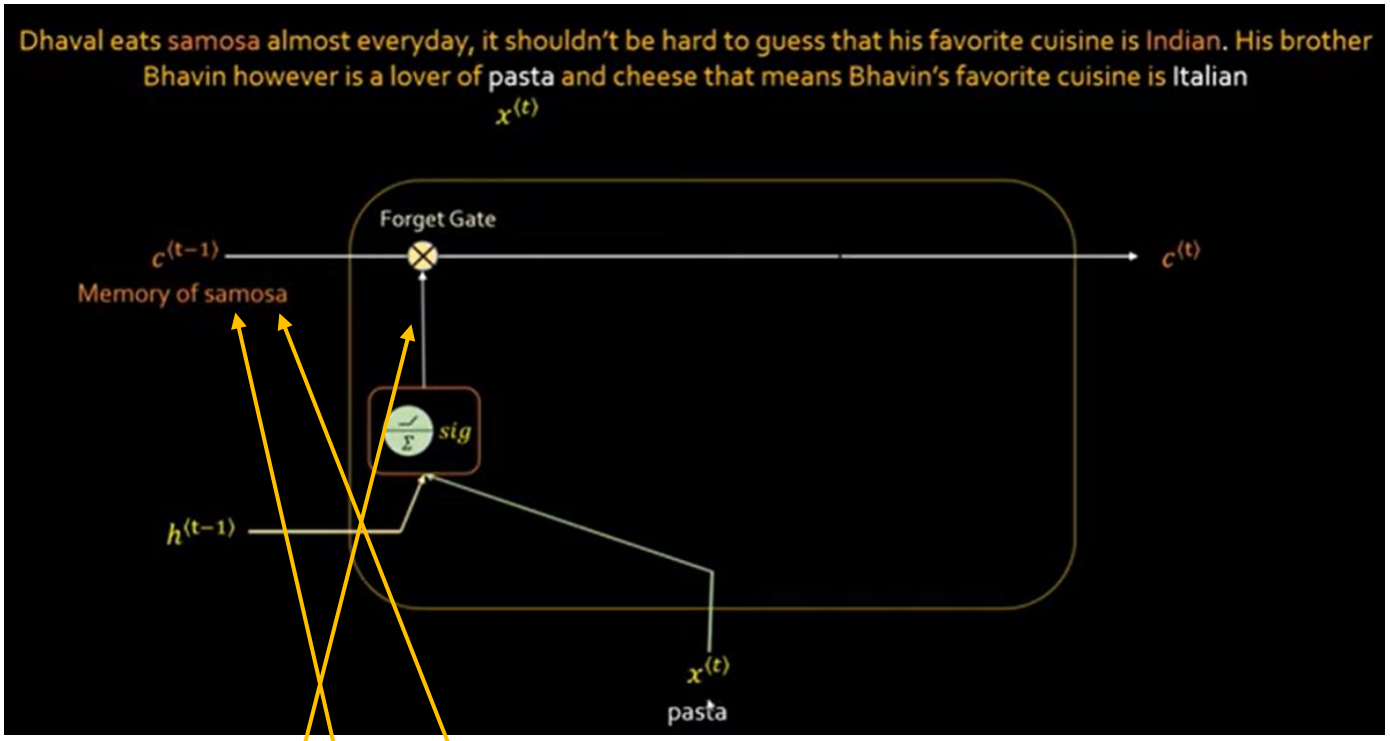
So when u r training ur RNN u r not only giving this particular statement ,this is a statement for prediction

When the training is happening u r giving 1000's and 1000's of such statements which will build that understanding in RNN on what to discard and what to store

When u r talking about LSTM , each of this cells are LSTM cells.

The first important thing is the forget gate

So the role of forget gate is when u come at this word pasta it knows that it has to discard samosa.



So this is how forget gate looks like

X(t) is a one word, u process sentence word by word t is the time stamp that's why x(t)

Forget gate is simple, u have previous hidden state

U take current input which is ur current word and u apply sigmoid function, u know that sigmoid function restricts ur number between 0 and 1

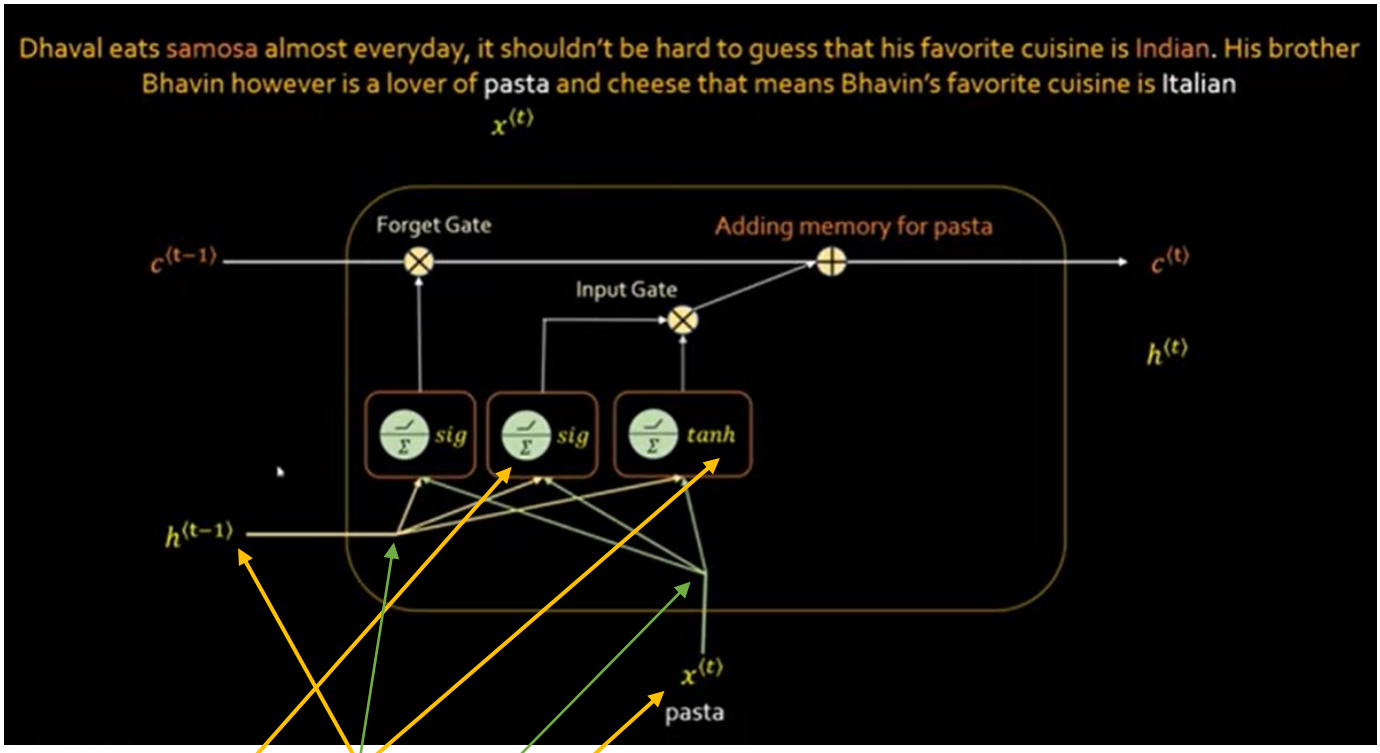
If sigmoid function has to discard the previous memory, sigmoid function will output a vector which will have all 0's or the values which are close to 0

And when u multiply that with previous memory which is previous cell state

U know u have a vector of all 0's here and u have another vector which is a memory of previous cell state

The multiplication will of course be 0, u have discarded the previous memory, when this new word appeared

This is what forget gate looks like, so here u forgot about samosa



There is another thing which is input gate

So when pasta came , not only u forgot about samosa, u need to add memory of pasta

U will use sigmoid and tanh on these two vectors

These vectors will have weights here

So in this function we are doing h(t-1) multiplied by its weight + x(t) multiplied by its weight + bias and then u r applying tanh on top of it

And it's the same equation here the only difference is instead of tanh u r using sigmoid function

And u multiply outputs of sigmoid function and tanh function and then add that as a memory for the word pasta x(t)

The third one is output gate