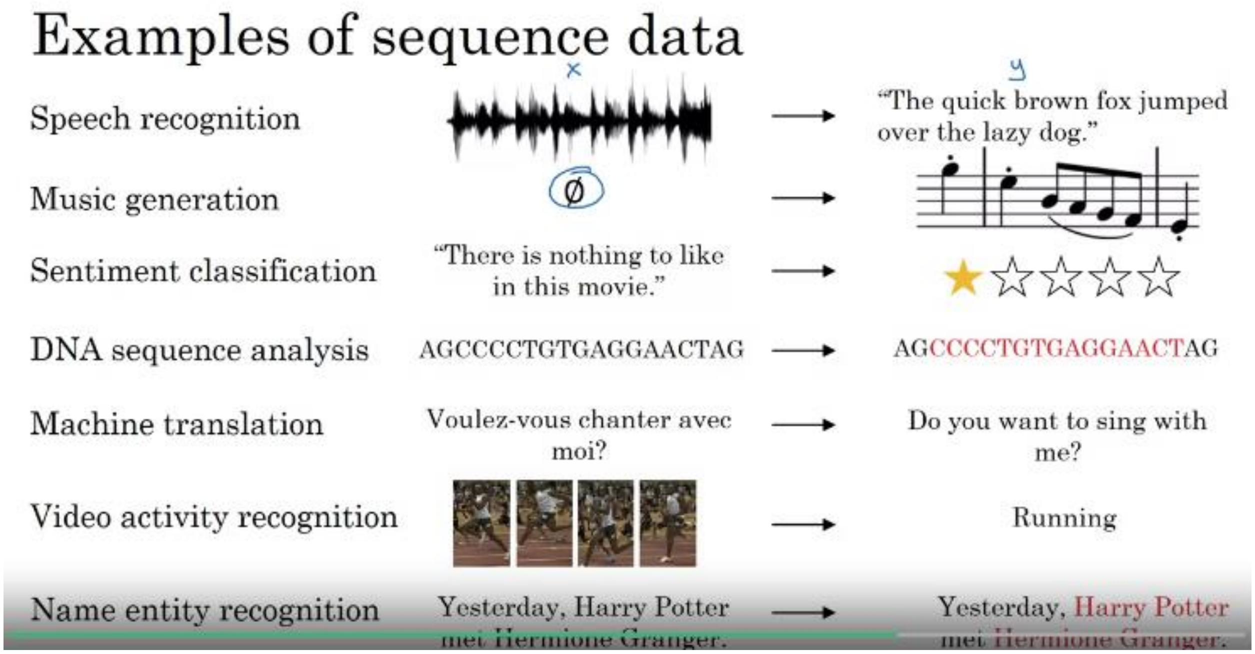
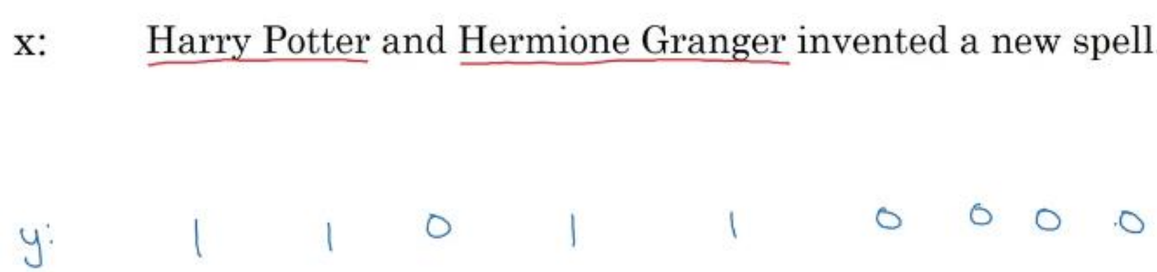
Why sequence models



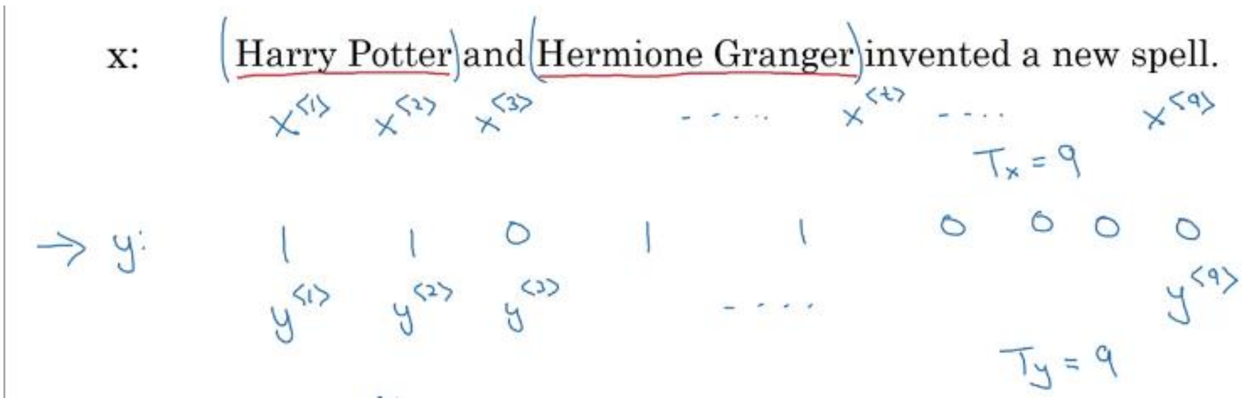
- Can be addressed as supervised learning problems
- In some, both the input X and the output Y are sequences, and in that case, sometimes X and Y can have different lengths,
- In some of these examples only either X or only the opposite Y is a sequence.

Named entity recognition: Identify names, times, locations etc in a sequence (used by search engines)
How would we frame X and y?



Maybe the above isn't the best output representation, there are some more sophisticated output representations that tells you not just whether a word is part of a person's name, but tells you where are the start and ends of people's names are in the sentence.

Now lets index the position of each word and its label, because these are temporal sequences.



Notation:

$x^{(i)}$: i^{th} training example

$x^{(i)<t>}$: t^{th} element of the sequence of training example i

$T_x^{(i)}$: : Length of the input sequence for training example i

$y^{(i)<t>}$: t^{th} element of output sequence of i^{th} training example

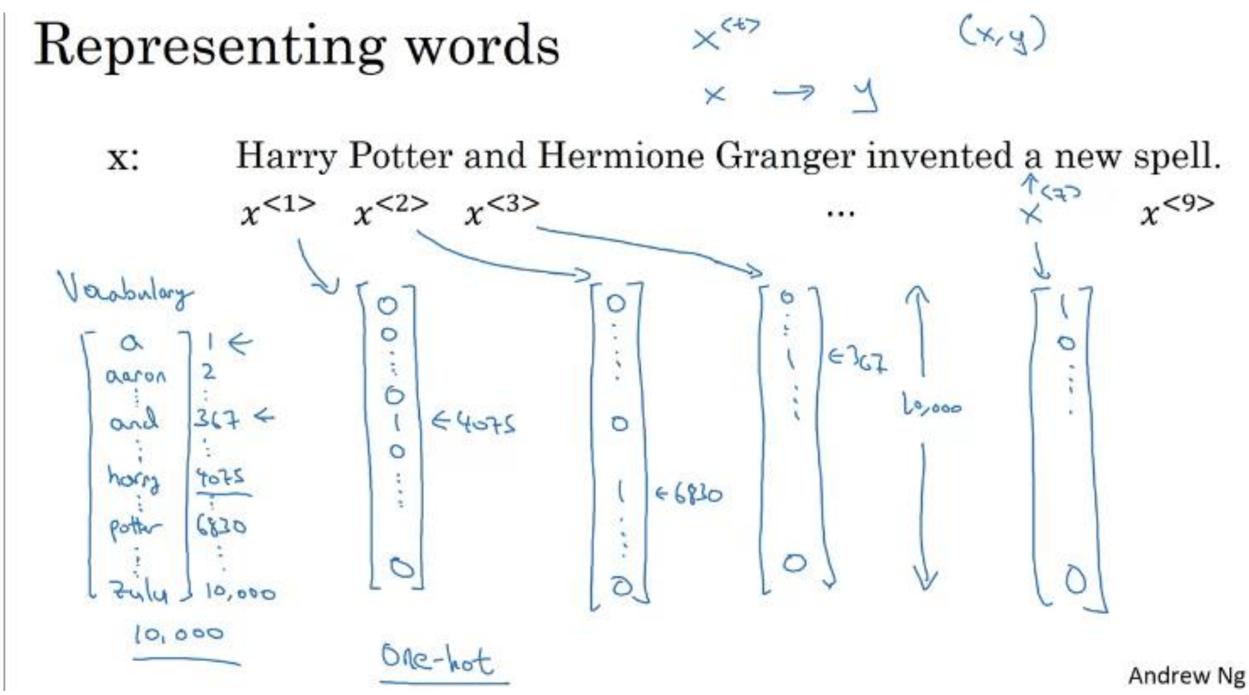
$T_y^{(i)}$: Length of output sequence of ith training example

In our above example, $T_x^{(i)}$ is 9 since we have 9 words

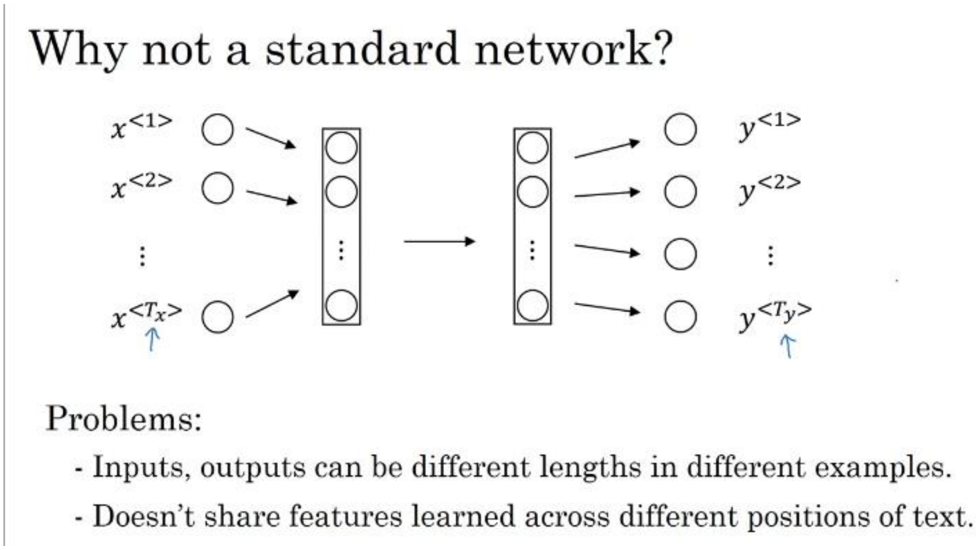
Representing words

We use a dictionary of vocabulary. e.g. lets consider a dictionary of 10,000 words

We'll use one-hot representation with this vocabulary



RNN

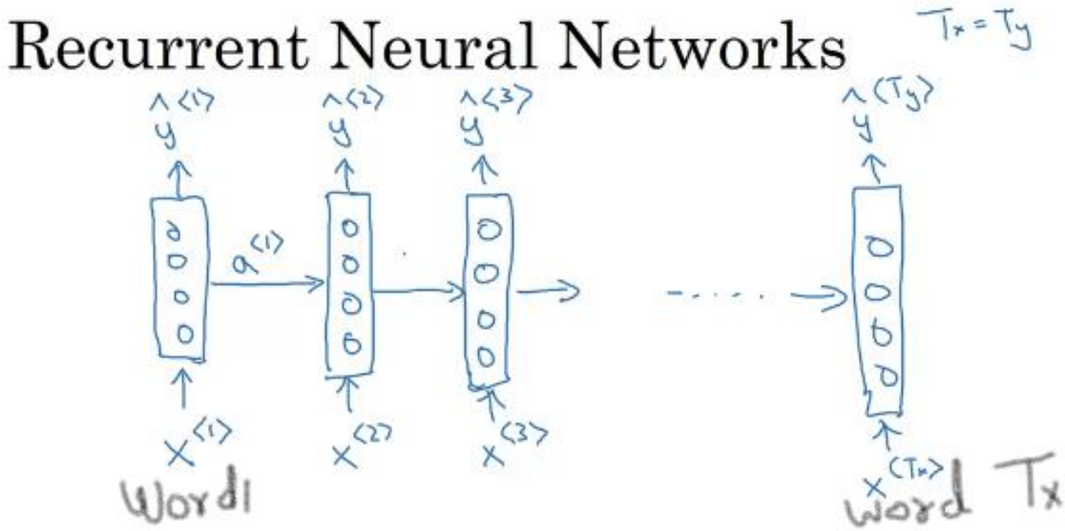


Problems:

1. Inputs and outputs can be different lengths for different examples. Maybe we could identify the maximum length of a sentence, and pad other smaller sentences upto that length, but this still wouldn't look like a good representation
2. Doesn't share features learned across different positions of texts. e.g. if the NN figures out that the word Harry appearing in position 1 is a sign that it is a person's name, it would be nice if it could automatically figure out that Harry appearing in another position is also a name. This is similar to CNN where we'd like things learnt in one part of an image to generalize well to other parts of the image as well.

An RNN addresses both these limitations

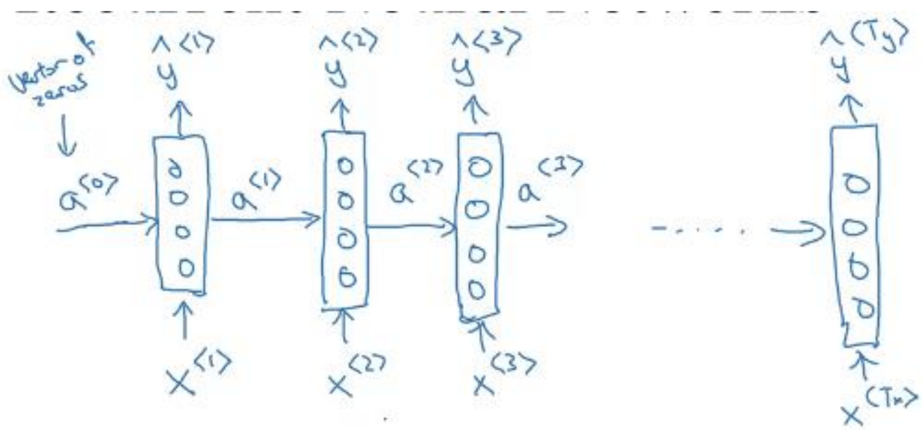
What is an RNN?



In the above, we first predict $y^{<1>}$ by feeding the first word $X^{<1>}$ into an NN layer. Then we read the second word $X^{<2>}$, and also use some information from what was computed at 1st step, to predict $y^{<2>}$.

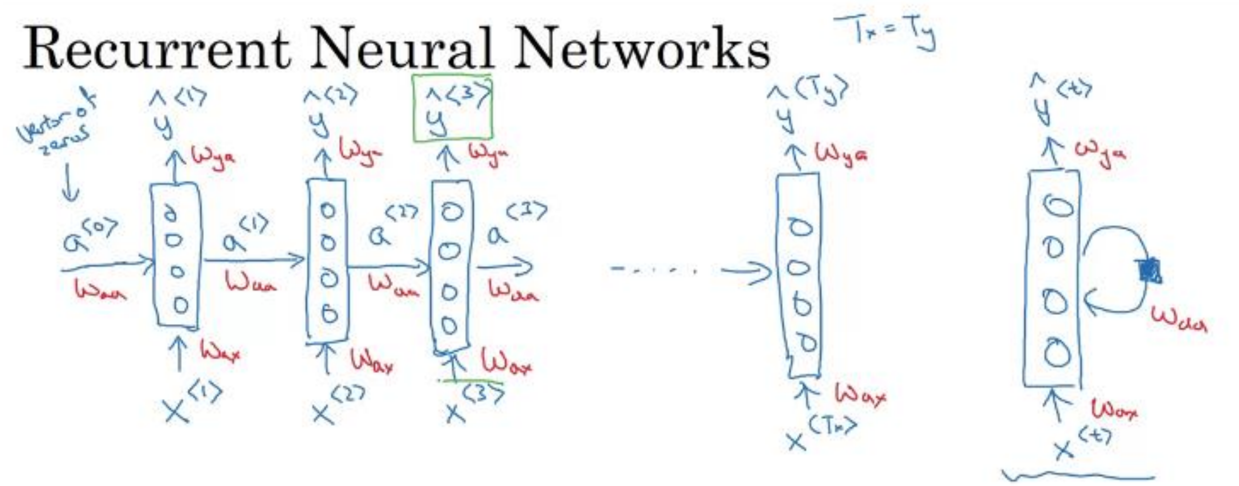
In this architecture, T_x and T_y are identical, i.e. input and output sequence lengths are the same.

We also add a vector of activations at time 0, usually a vector of 0s.

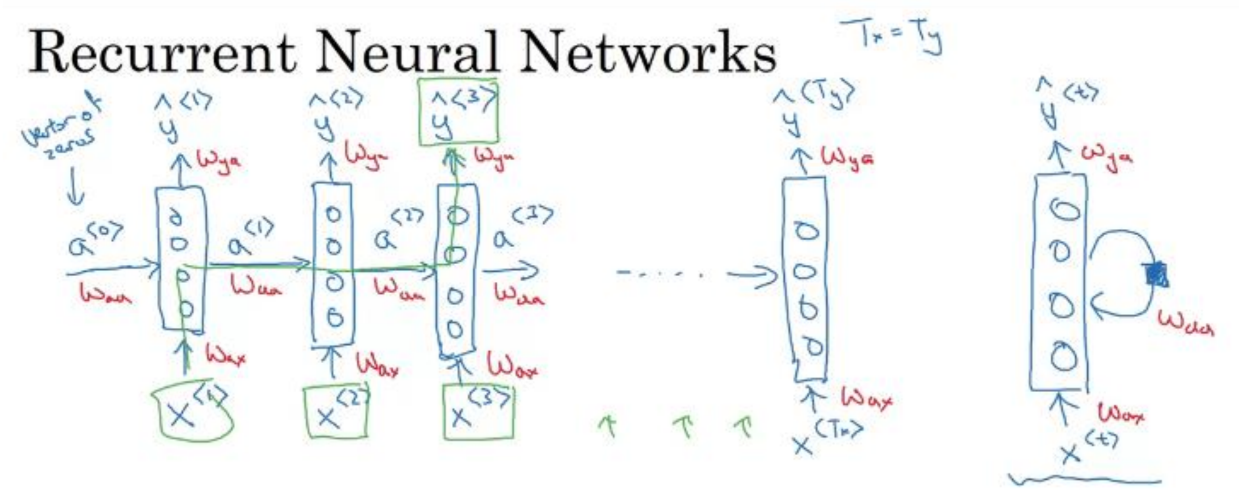


At every time step:

1. W_{ax} (parameters governing connection from X to hidden layer) are identical
2. W_{aa} (parameters governing horizontal connections) are identical
3. W_{ya} (parameters governing output predictions) are identical



So prediction $y^{<3>}$ uses info from not only $X^{<3>}$ but also $X^{<1>}$ and $X^{<2>}$



Drawback: Doesn't use $X^{<4>}$, $X^{<5>}$ etc i.e. later words in the sentence.

In below sentence, not possibly to decide whether Teddy is part of a person's name by **just** looking at the first three words.

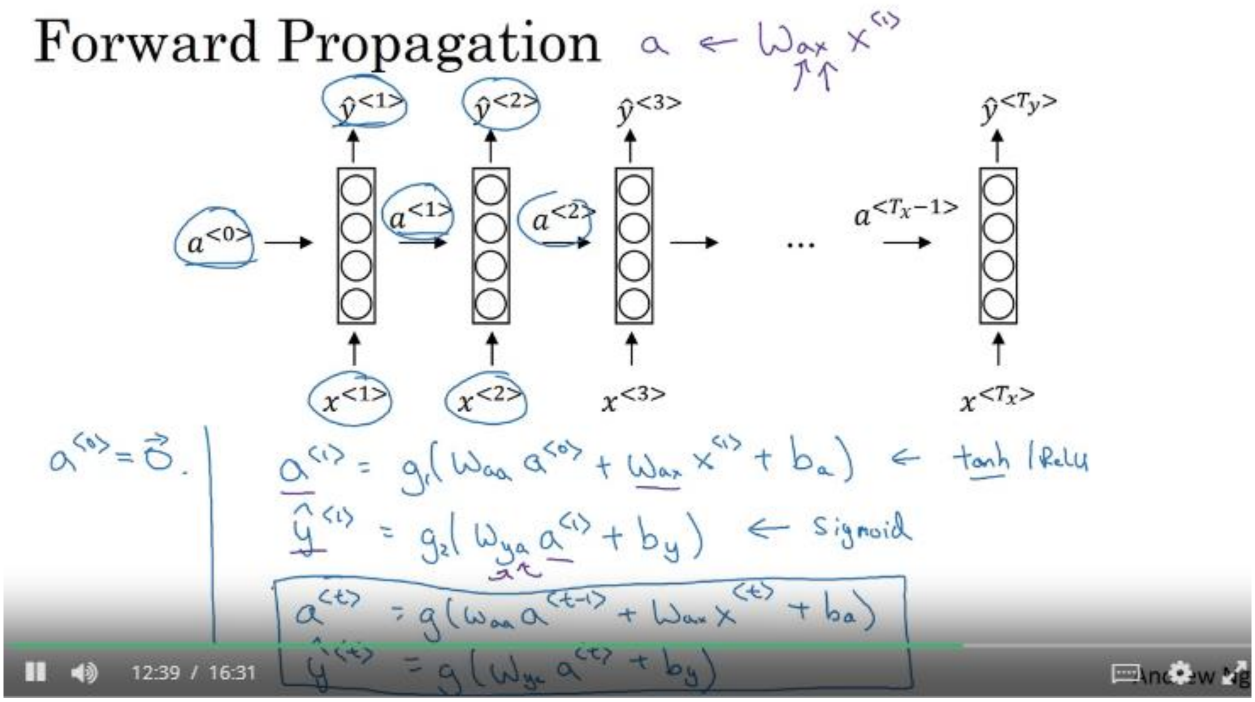
He said, "Teddy Roosevelt was a great President."

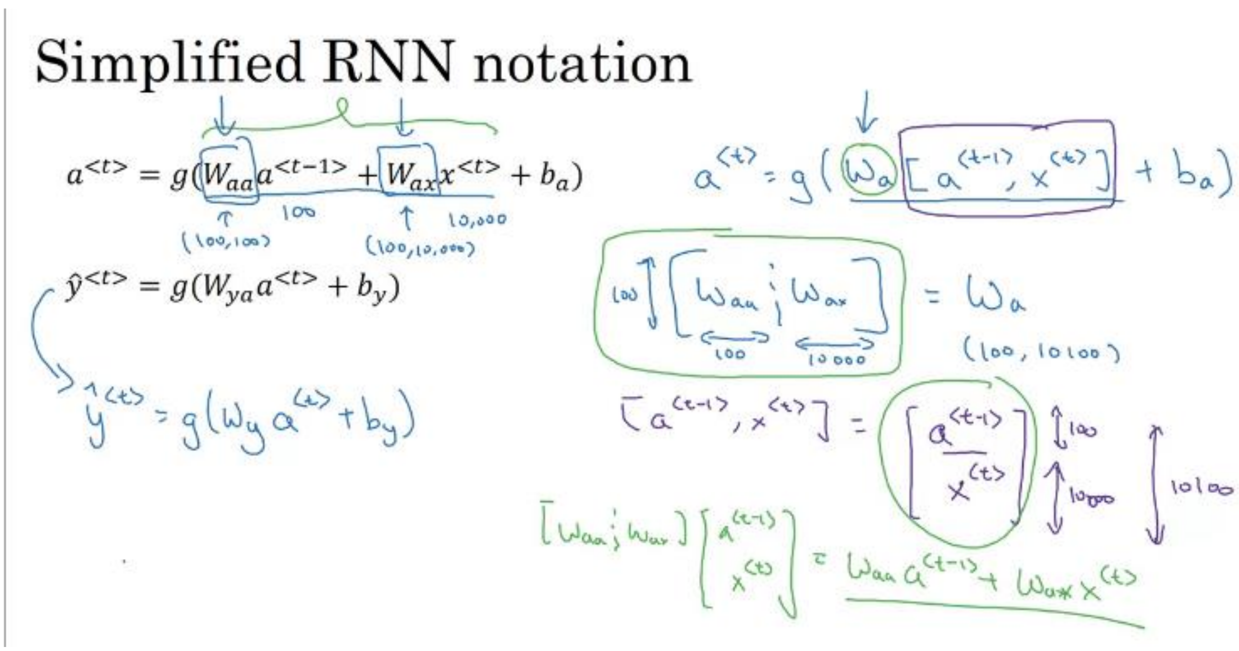
He said, "Teddy bears are on sale!"

Andrew Ng

Later in this course: Bidirectional RNNs (BRNNs) address this

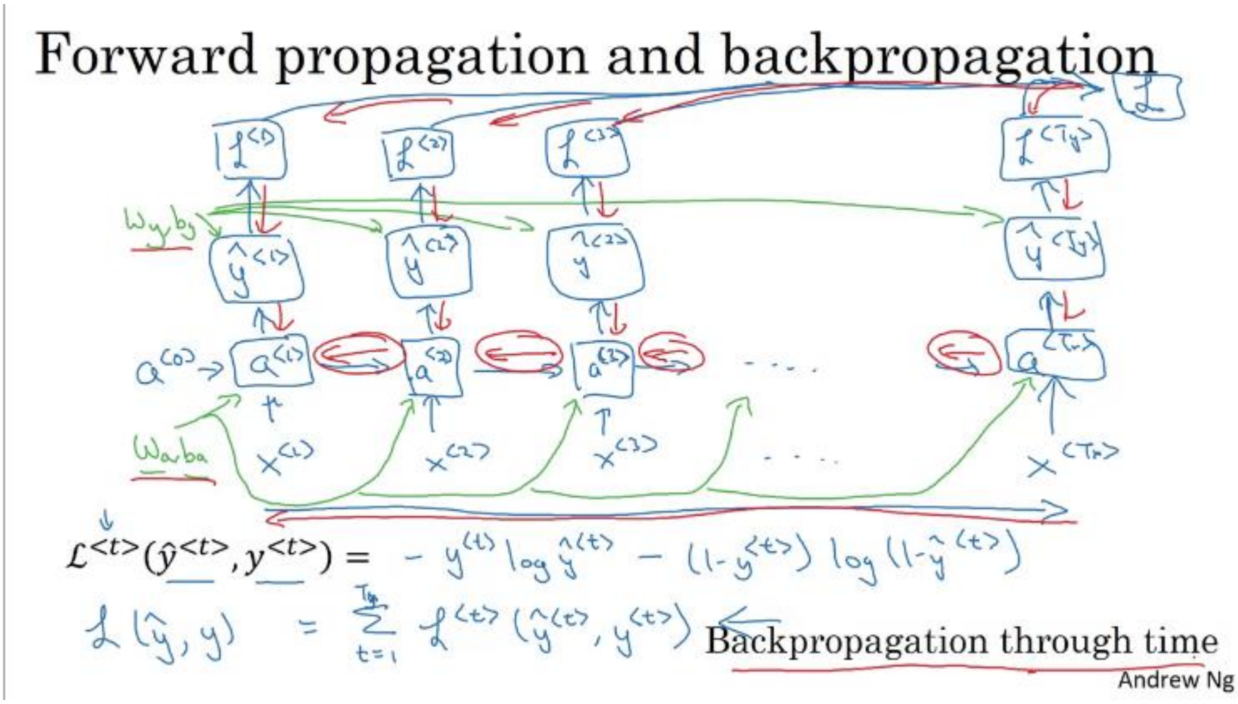
Forward propagation



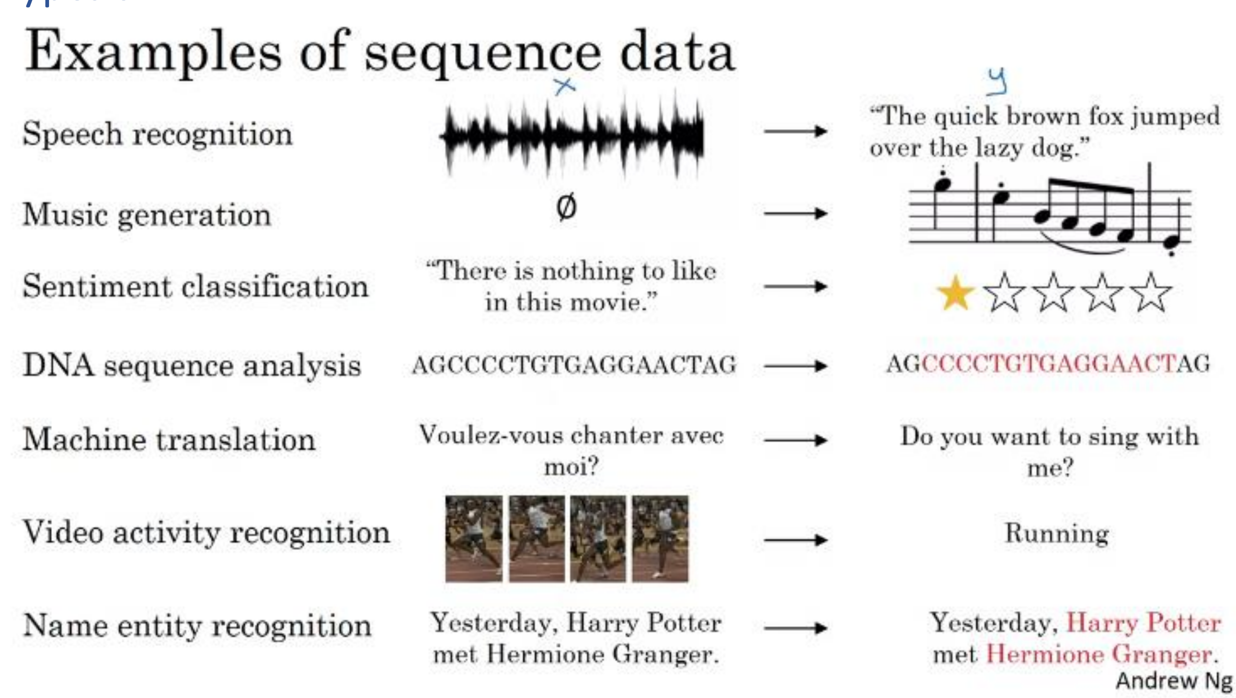


See my handwritten notes for the math

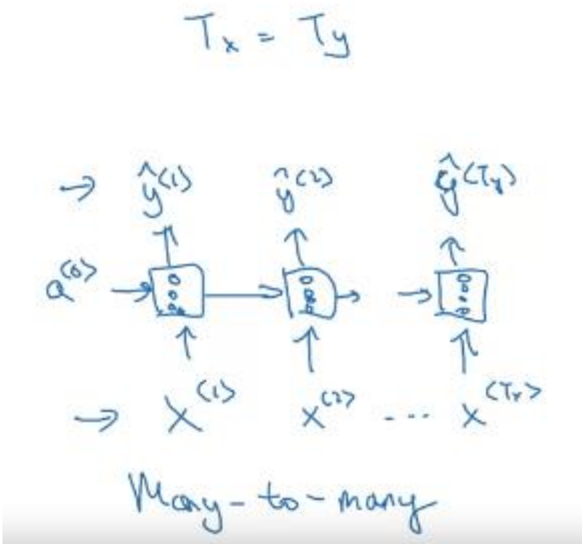
Backpropagation through Time



Types of RNN

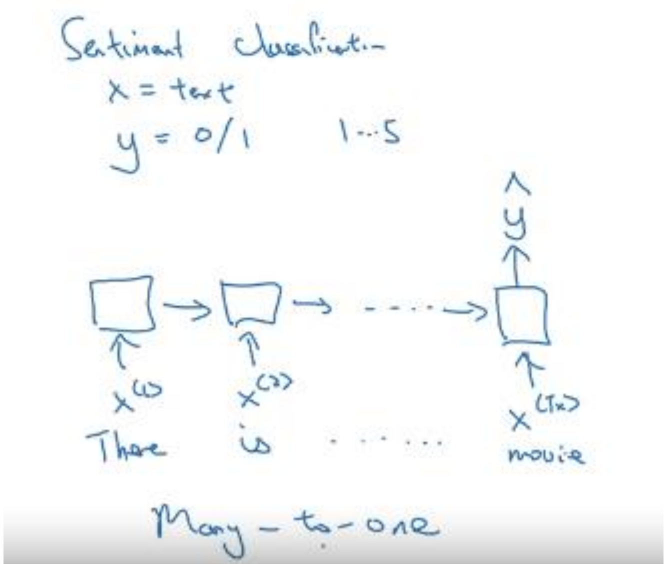


Many to many

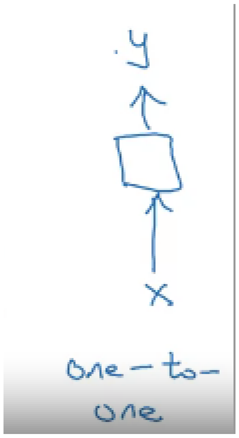


Many to one

Sentiment classification

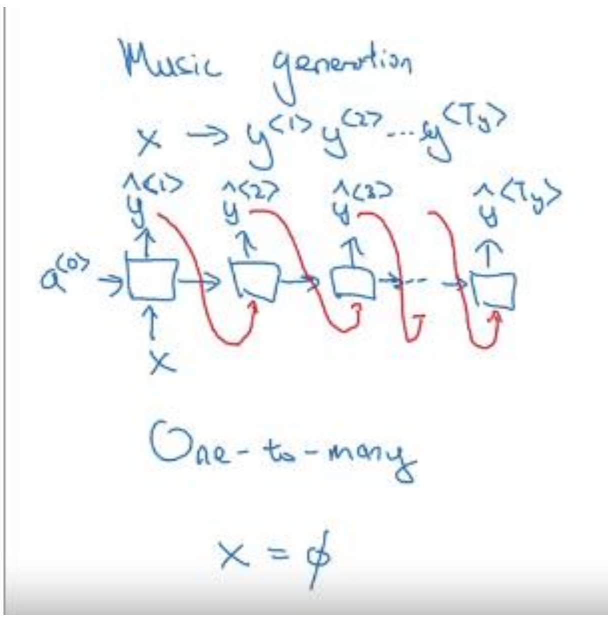


One to one



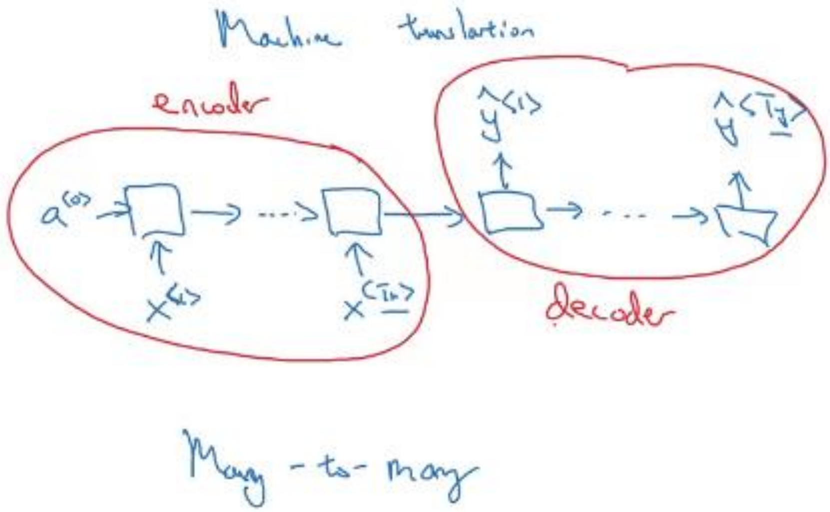
One to many

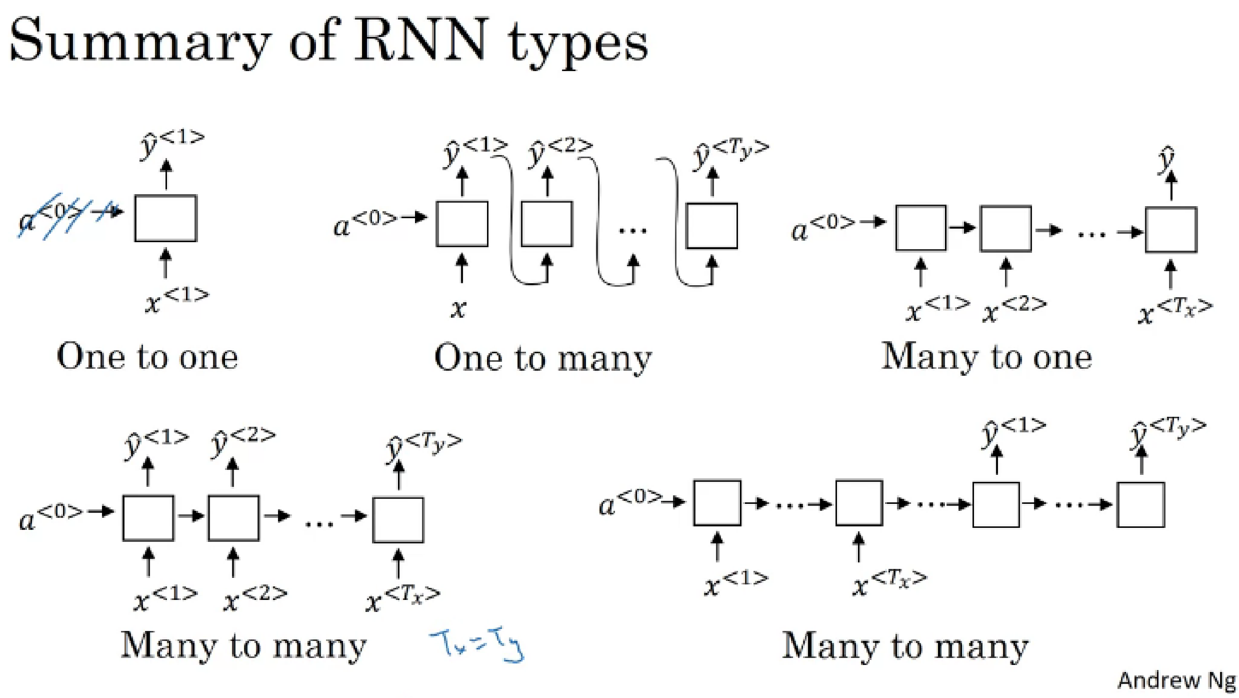
Music generation



Many to many with different input and output lengths

e.g. machine translation





Language model & sequence generation

Estimates the probability of that particular sequence of words.

What is language modelling?

Speech recognition

The apple and pair salad.

The apple and pear salad.

P(The apple and pair salad) = 3.2×10^{-13}

P(The apple and pear salad) = 5.7×10^{-10}

P(sentence) = ?

$P(y^{<1>}, y^{<2>}, \ldots, y^{<T_y>})$

First we tokenize the sentence, and one-hot encode

We also add a token for end-of-sentence

Language modelling with an RNN

Training set: large corpus of english text.

Tokenize

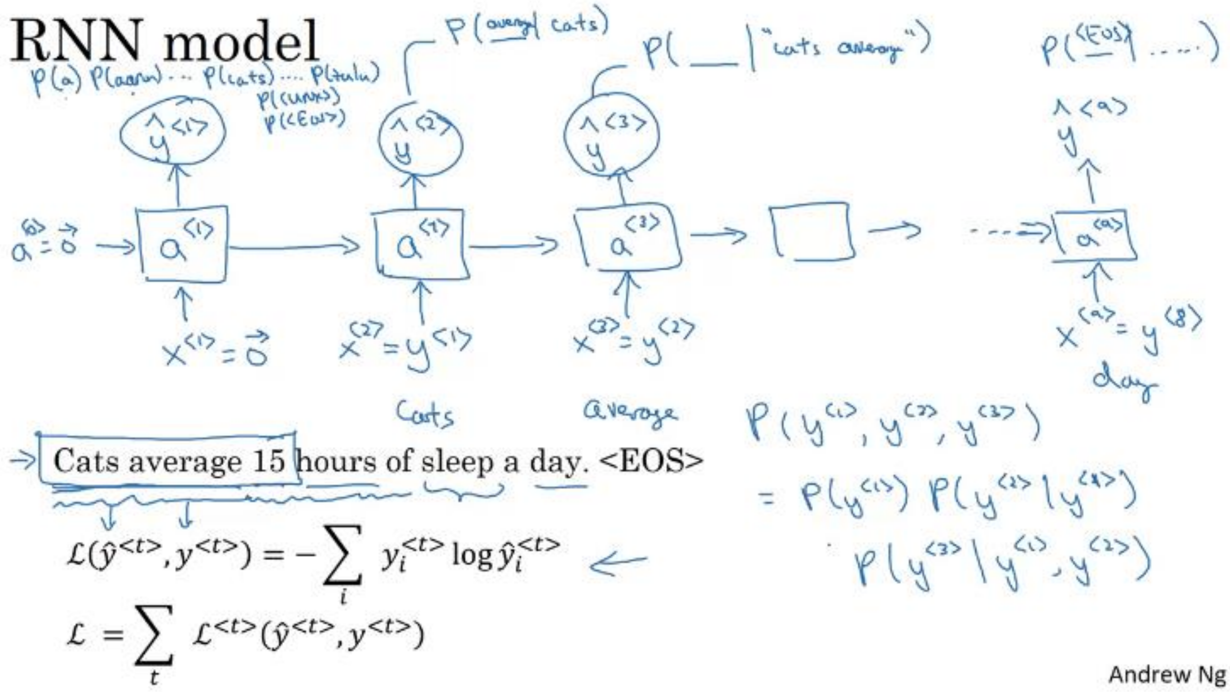
Cats average 15 hours of sleep a day. <EOS>

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $\ldots$ $y^{<8>}$ $y^{<9>}$

The Egyptian ~~Mau~~ is a bread of cat. <EOS>

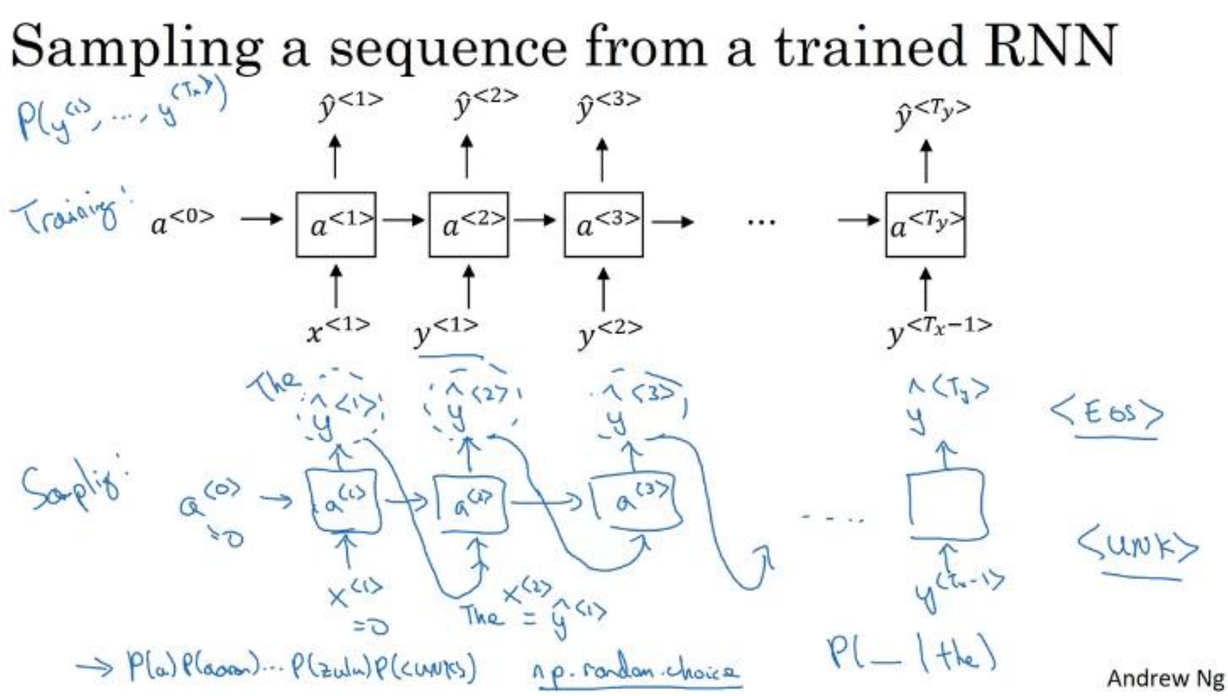
10,000 <UNK>

Andrew Ng



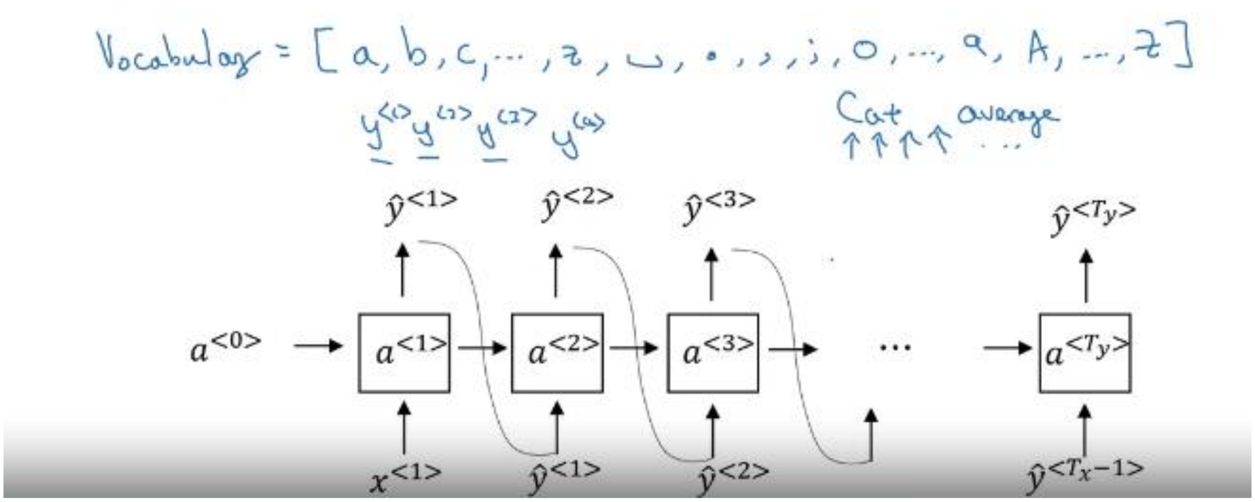
Sampling novel sequences

Sequence models model the chance of any particular sequence of words, so what we like to do is sample from this distribution to generate novel sequences of words.
At the first timestep, we sample from the vocabulary of our words using softmax function.
At the second timestep, we pass in the sampled word from the first timestep as an input i.e. $x^{<2>}$ = yhat<1>.
And so on till end of all timesteps.



We can also build a character level RNN

Character-level language model



Pros:

1. Don't have to worry about unknown word tokens

Cons

1. End up with much longer sequences
2. Computationally expensive

Sequence generation

News

President enrique peña nieto, announced sench's sulk former coming football langston paring.

"I was not at all surprised," said hich langston.

"Concussion epidemic", to be examined.

The gray football the told some and this has on the uefa icon, should money as.

Shakespeare

The mortal moon hath her eclipse in love.

And subject of this thou art another this fold.

When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

Vanishing gradients with RNN

We should match 'cats' with 'were' and 'cat' with 'was'.

But basic RNNs are not good at capturing long term dependency.

The $\boxed{\text{cat}}$, which already ate ...., $\boxed{\text{was}}$ full
The $\boxed{\text{cats}}$, wh . . . ., $\boxed{\text{were}}$ full

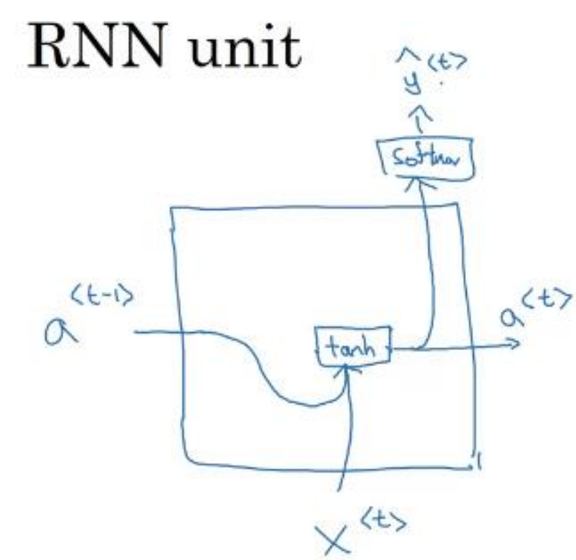
Outputs have local influences, ie values closer in the sequence

Recall that for exploding gradients, we use gradient clipping.

Gated Recurrent Units (GRU)

Helps capture long term dependencies

Below is the visualization of the RNN unit of the hidden layer of the RNN



$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

GRU units will have a new variable C which is a memory cell

At every timestep, $\tilde{C}^{<t>}$ will be a candidate value for replacing the memory cell

C = memory cell

$$c^{<t>} = a^{<t>}$$

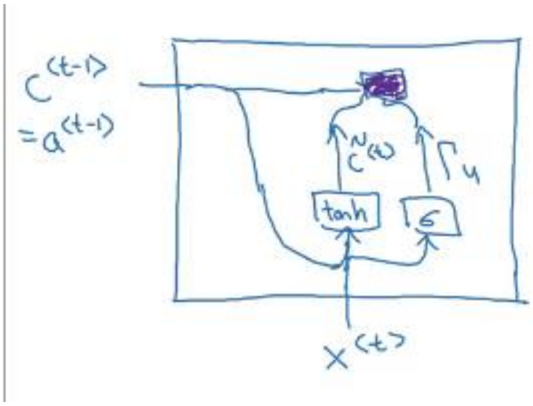
$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

The important idea of GRU is a gate, Γ_u where the u refers to update. Can think of it as always 0 or 1, though computed by sigmoid function.

Gate

The new $C^{<t>}$ will be Γ_u times the candidate value, plus 1- Γ_u times the old value.

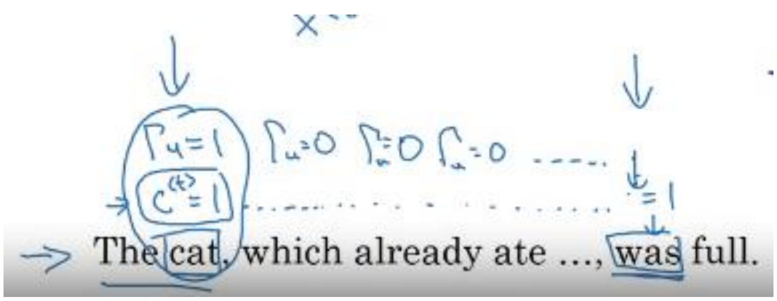
So when Γ_u is 1, we update to candidate value. If 0, then don't update, hang on to old value.



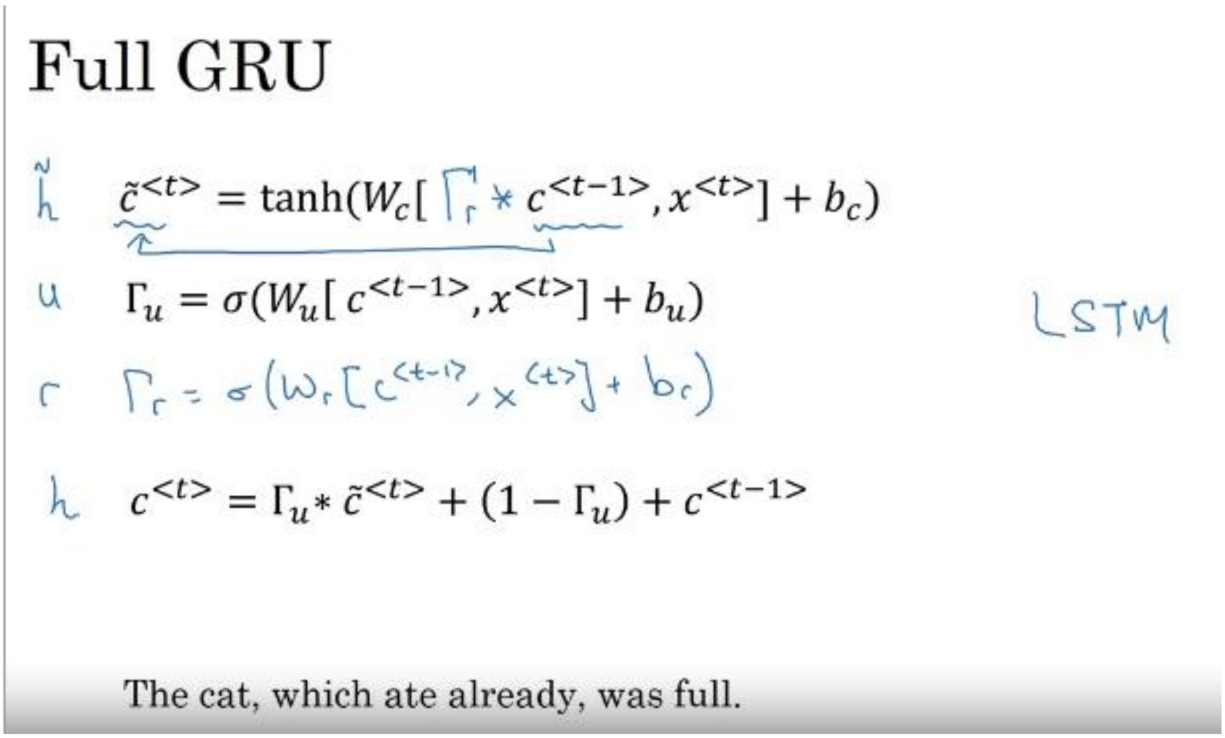
$$c^{<t>} = \underbrace{\Gamma_u}_{=1} * \tilde{c}^{<t>} + \underbrace{(1-\Gamma_u)}_{} * c^{<t-1>}$$

The purple box above is just

So the word 'cat' is read early in the sentence, so the update gate is set to , and then is not updated at successive. Finally it gets used to decide the word "was"



With tiny value of Γ_u, the previous value of C_t basically continues to persist without any update.


Full GRU
$\tilde{h}$ $\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$
u $\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$
LSTM
r $\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$
h $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$
The cat, which ate already, was full.

LSTM