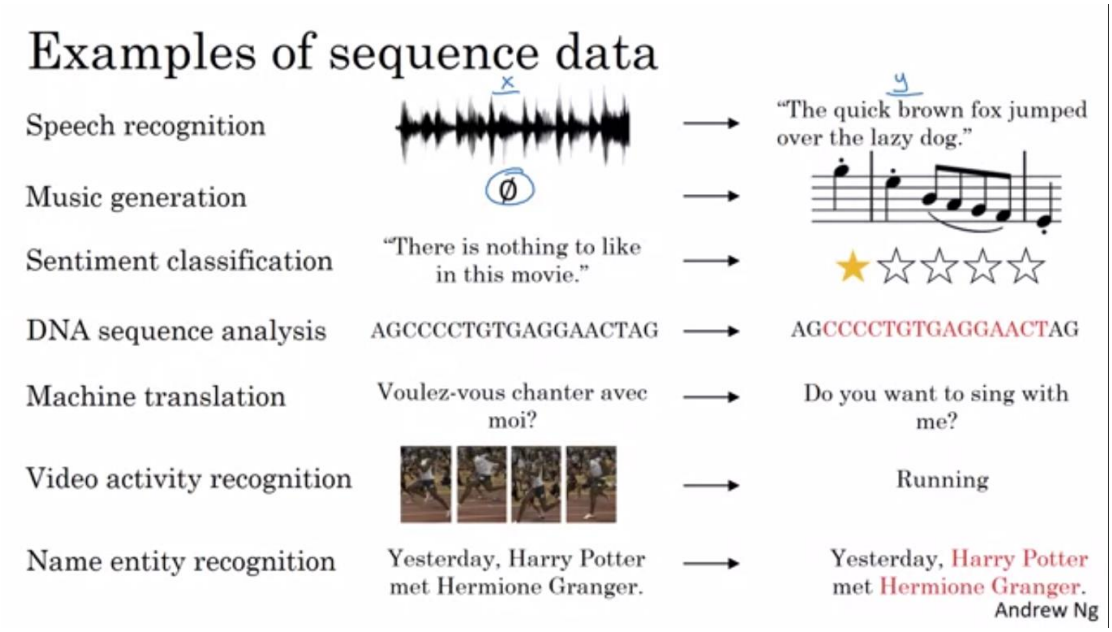
Ahmad Hussameldin Hamed Hassan

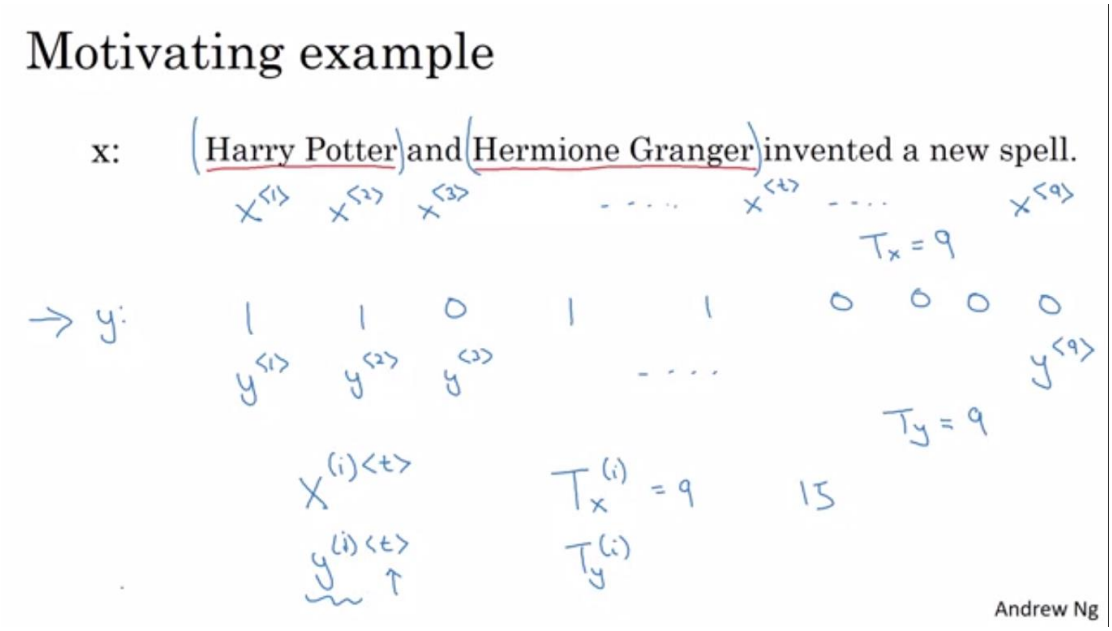
Shared Git-hub link: https://github.com/ahmadhassan1993/sharing-github

Recurrent Neural Networks (RNN)

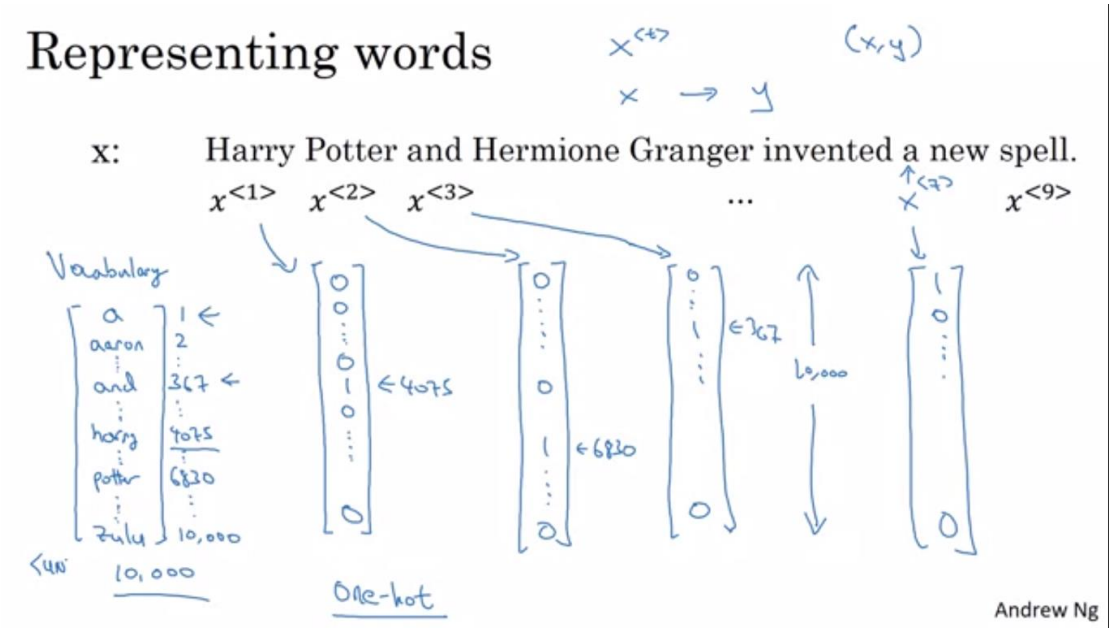
Input and output may both be sequences, like speech recognition or only output is sequence, like Music generation. We may also identify a sequence from a given sequence, like protein from DNA sequence and name identity recognition. Both inputs and outputs may have same length or different.



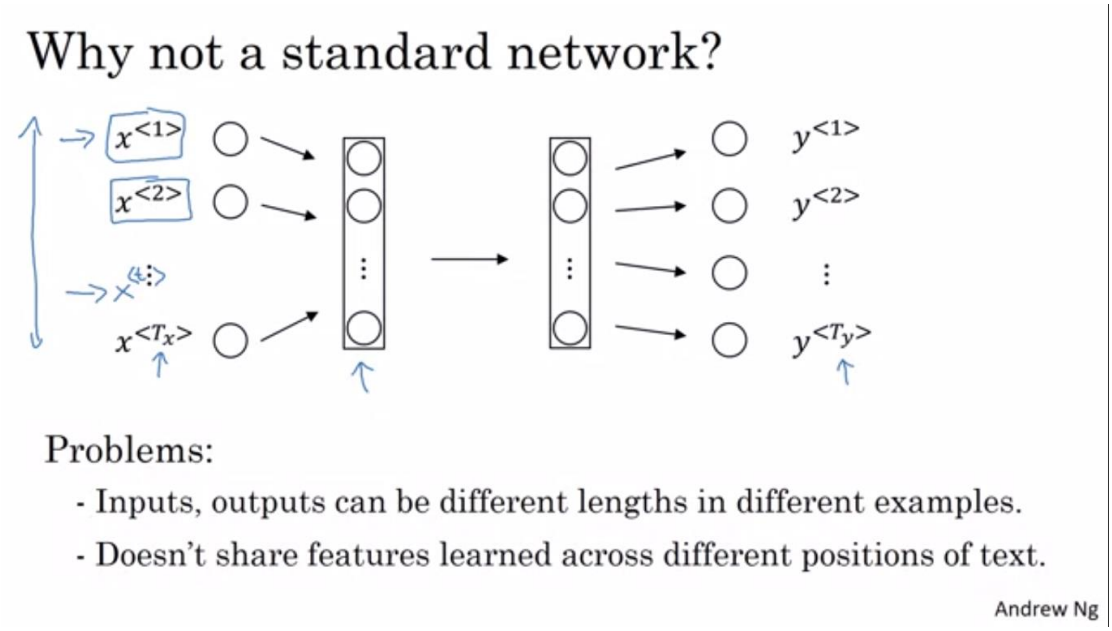
$x^{<t>}$ and $y^{<t>}$ represent the position of the item in the input and output sequences, respectively. Tx and Ty represent the length of the sequence for both input x and output y.



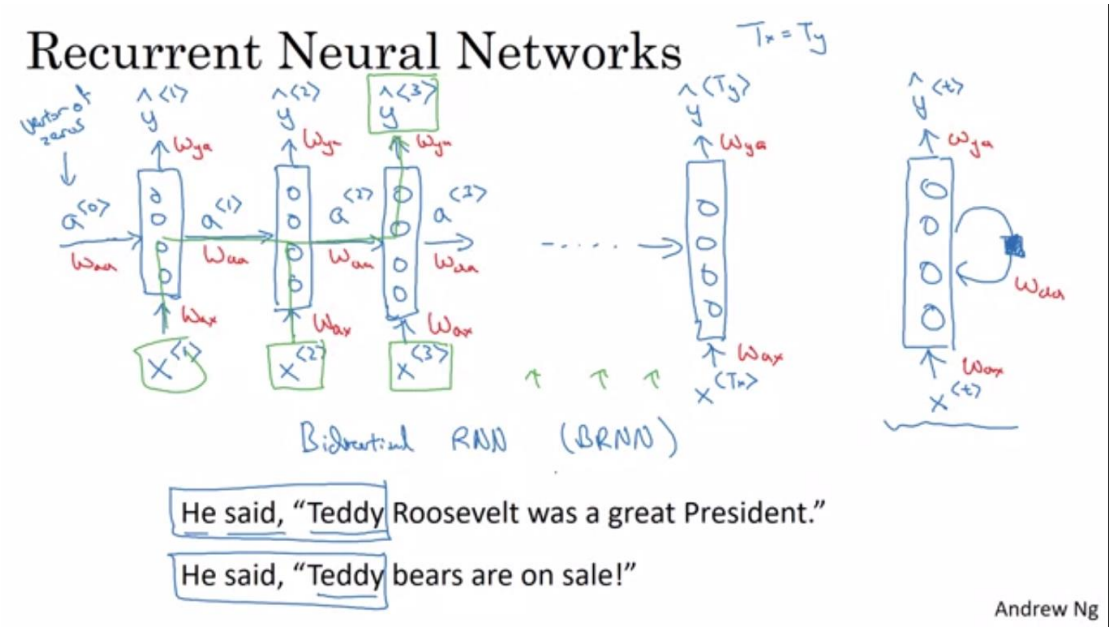
In Natural Language Processing (NLP), we represent a word in a sentence by one-hot vector $x^{<t>}$. The hot one location in the vector represent the word from our Vocabulary dictionary vector.



We can't use standard NN, because the inputs and outputs are varied for each example and feature learning is not generalized across network from single input (repetition will occur).

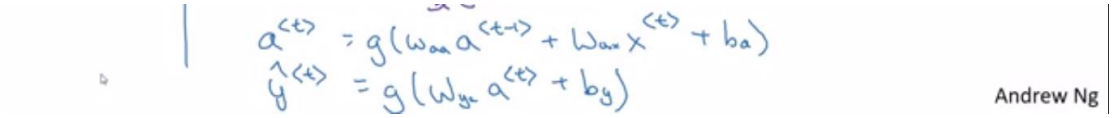


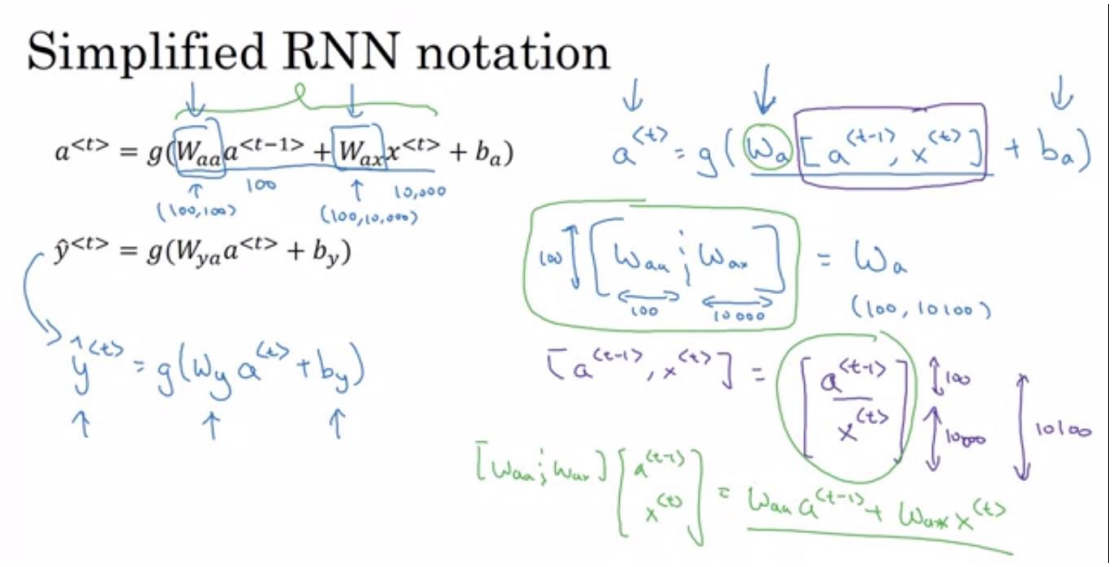
So, Recurrent NN (RNN) is the solution:



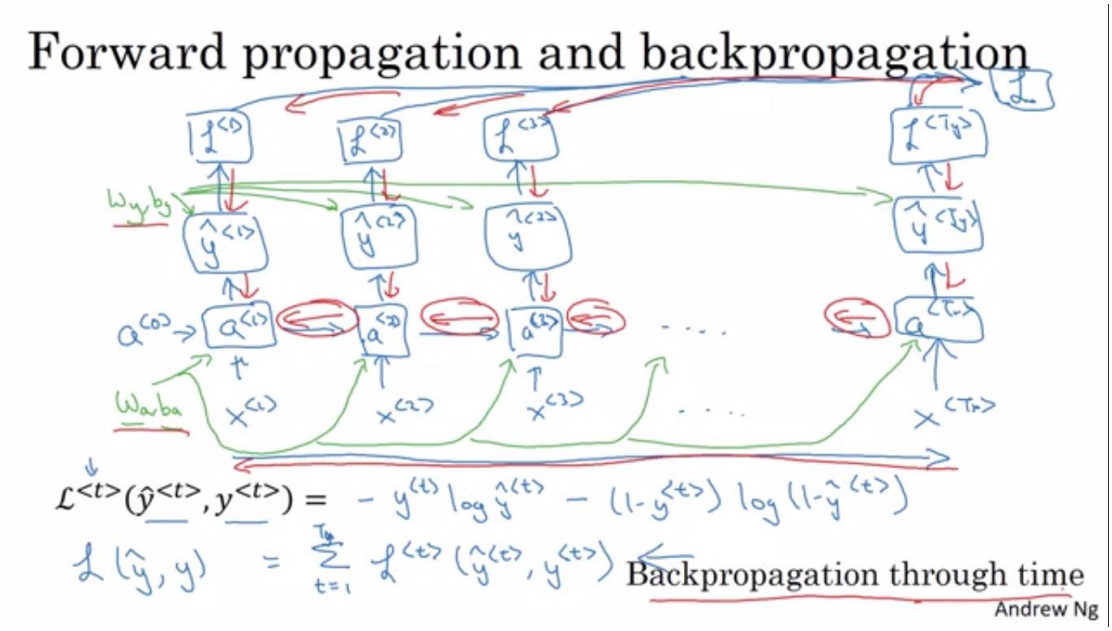
The parameters in red are same for all sequences process of each word. Bidirectional RNN is the advanced structure where every word sequence contributes in every word prediction, not only previous sequences. $a^{<t>}$ is the activation. a<0> is zeros.

w_{ax} and b_a are parameters to calculate a, while w_{ya} and b_y are parameters to calculate y^.



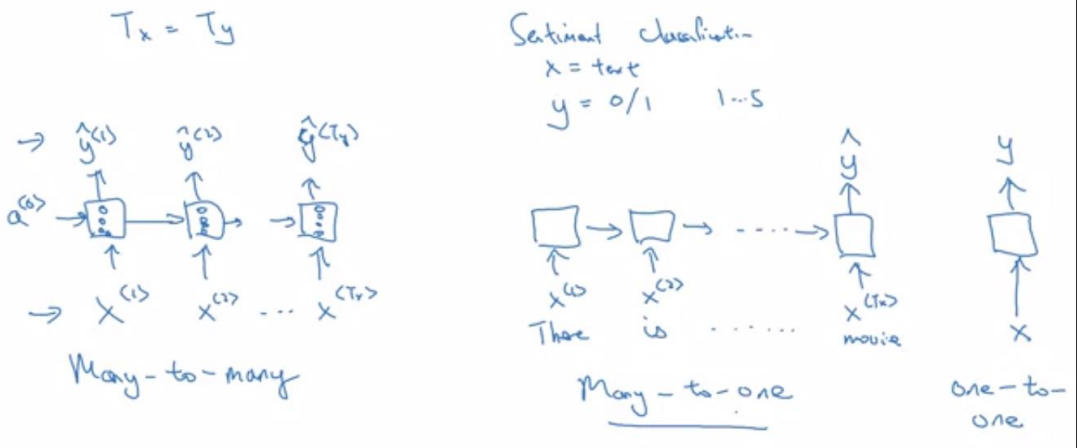


w_a is concatenation of w_{aa} and wax and w_y is simplified notation of way.



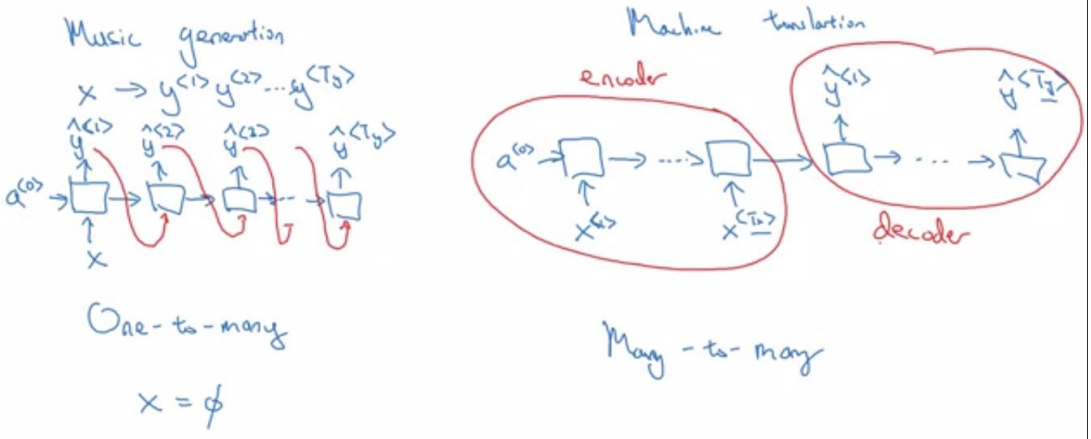
The previous RNN is many-to-many and equal input-output lengths. Other RNN types are: Many-to-one, one-to-one, one-to-many and many-to-many(with different input-output lengths).

Examples of RNN architectures

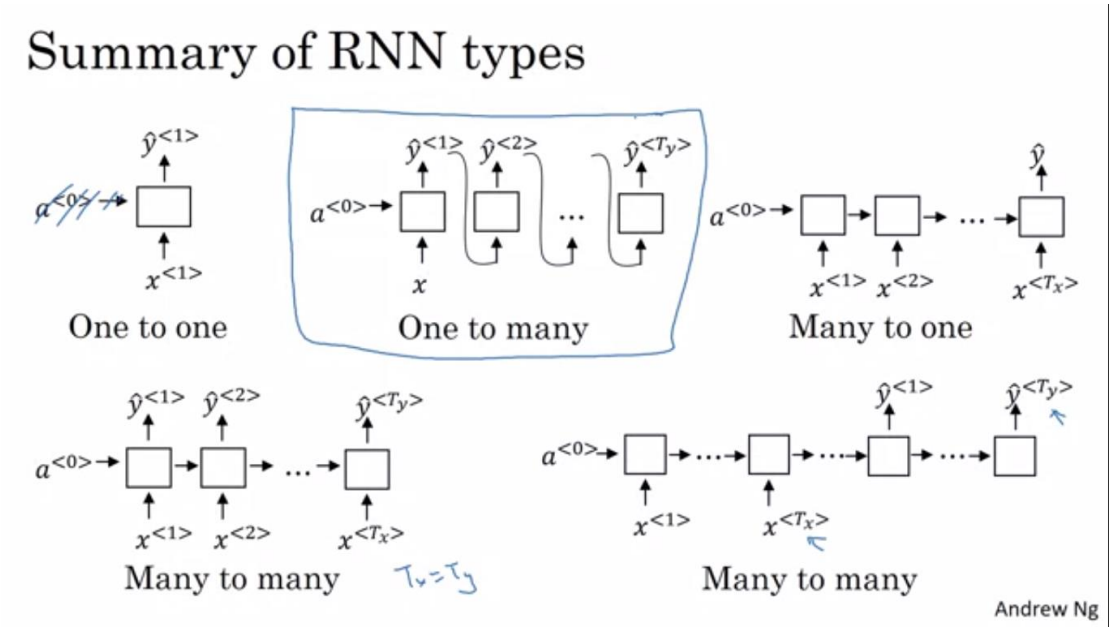


Andrew Ng

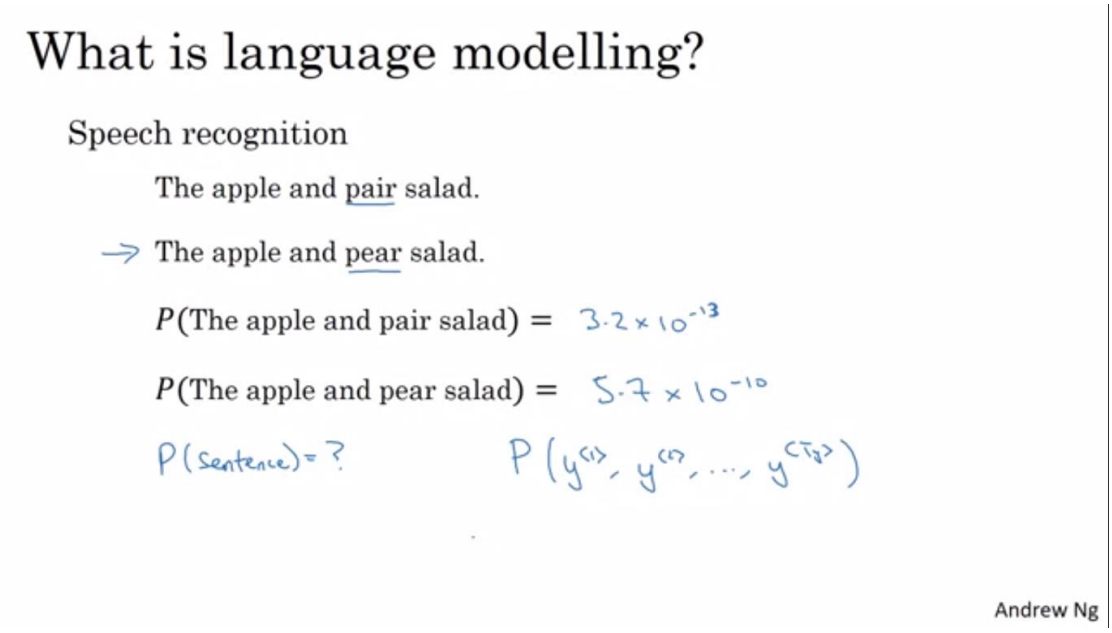
Examples of RNN architectures

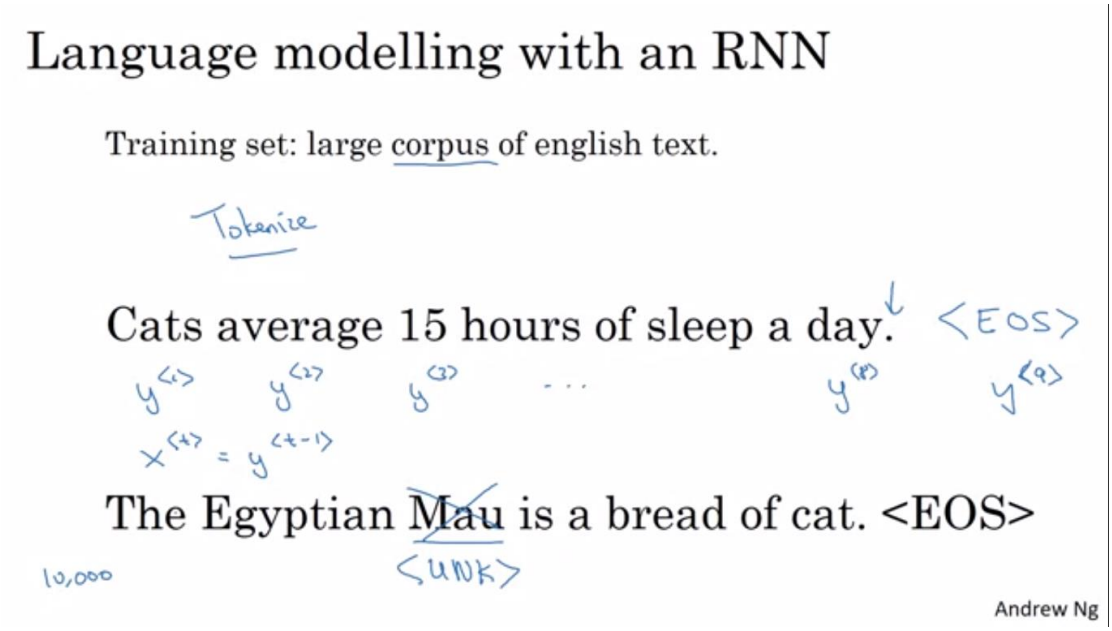


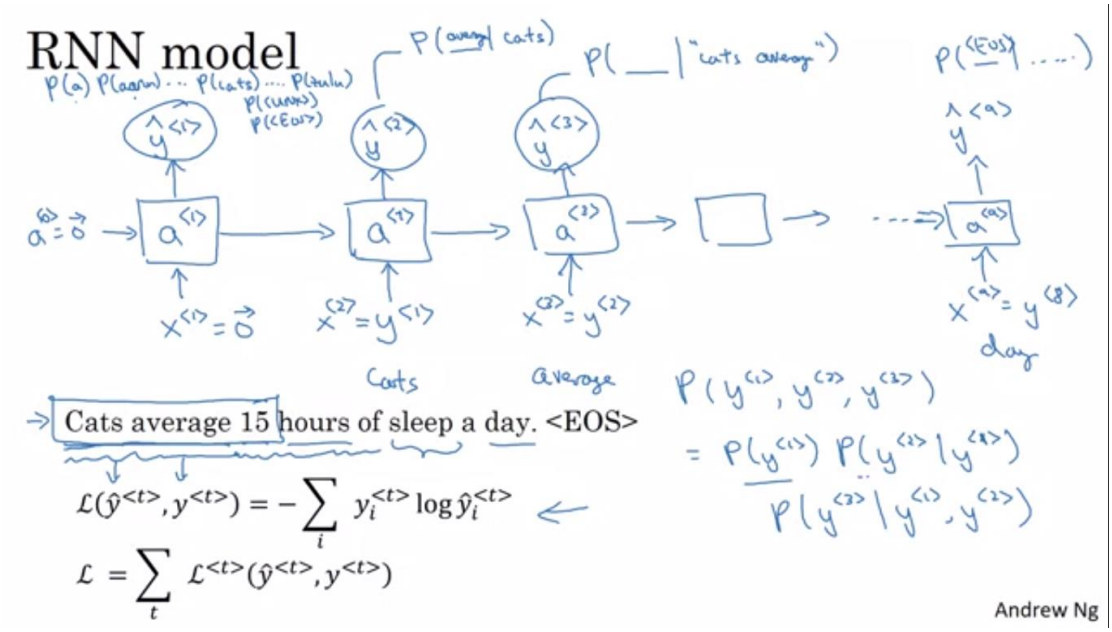
Andrew Ng



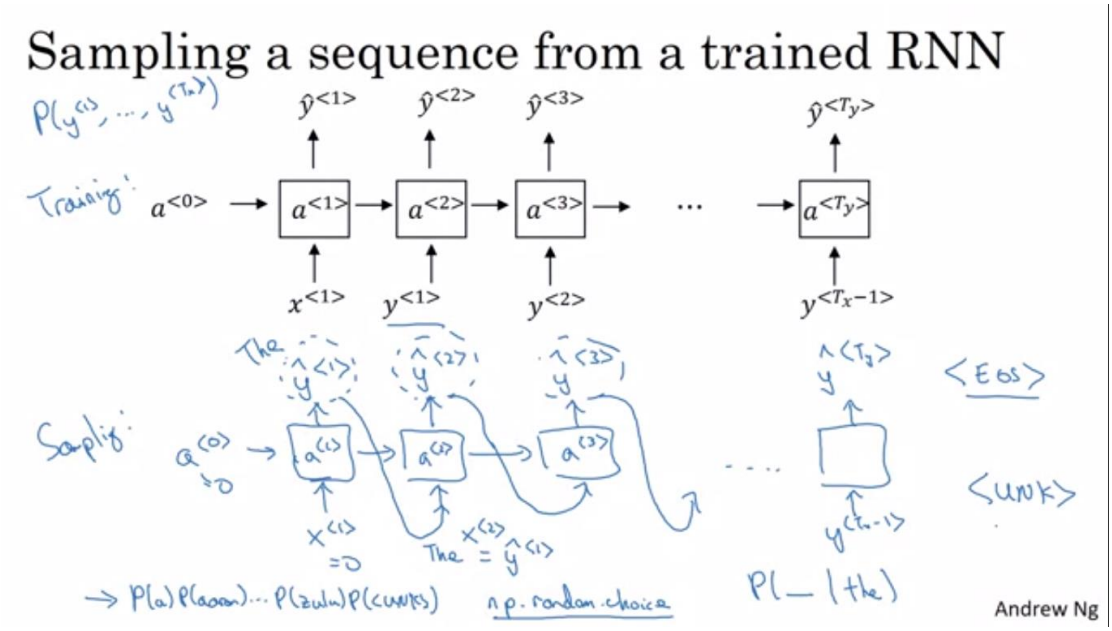
Language modelling is the probability that output sentence, for example, is correct in sense. We Tokenise the output to sequence of words. This will predict next word from given previous words in the sentence.



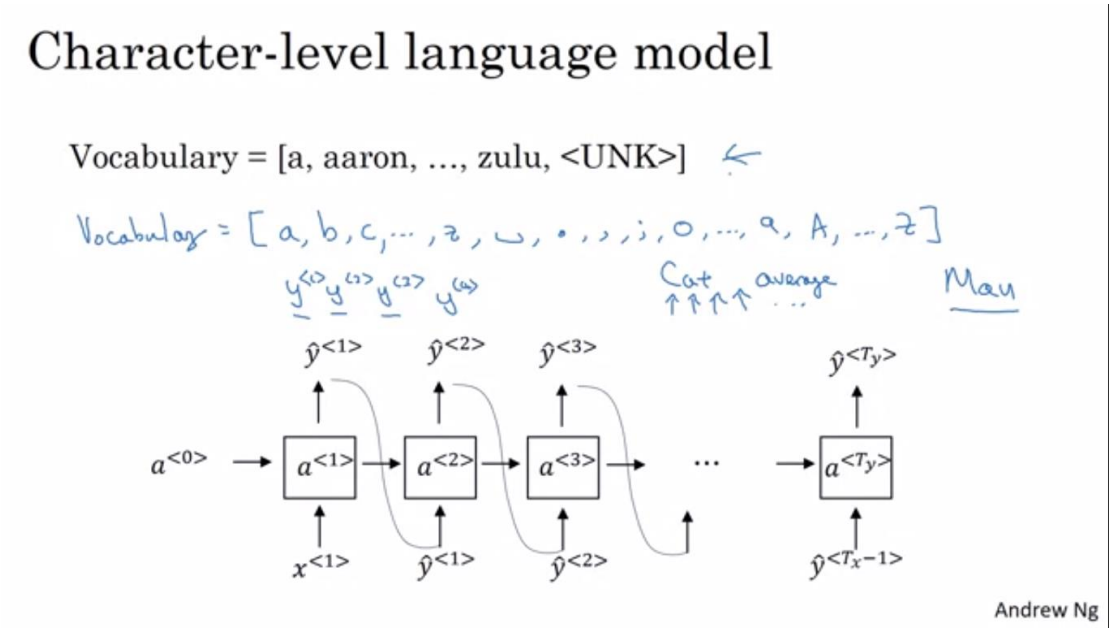




Sampling is application on language modelling that were used in training. It is generate a sequence from a given word.



Another language modelling is the character-level with high computation cost and less large end dependencies; how last part of the sentence agrees with first one.



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.

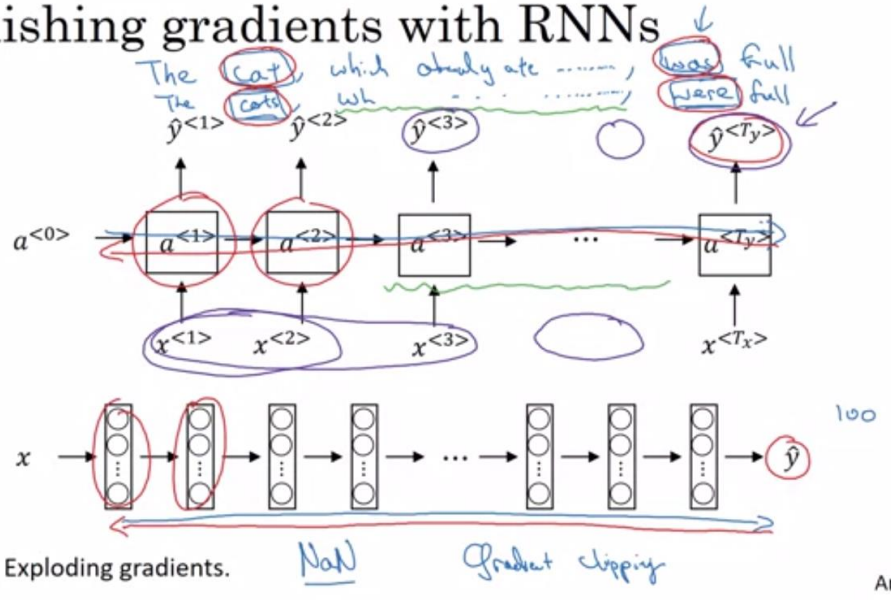
When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

Andrew Ng

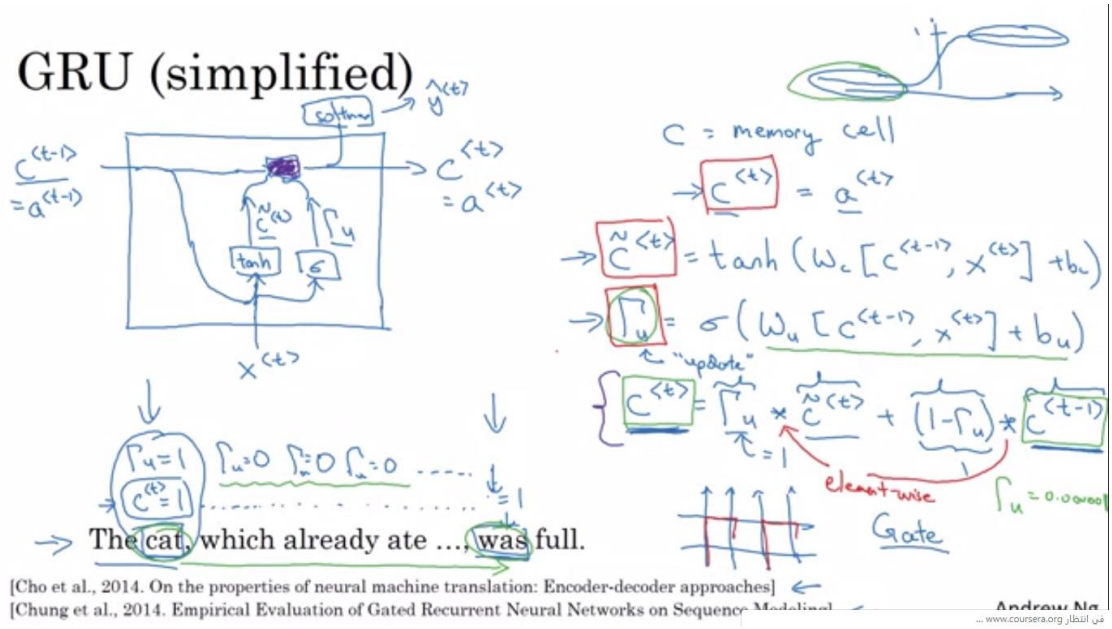
Larg term dependency in large DNN will lead to gradient vanishing or exploding.

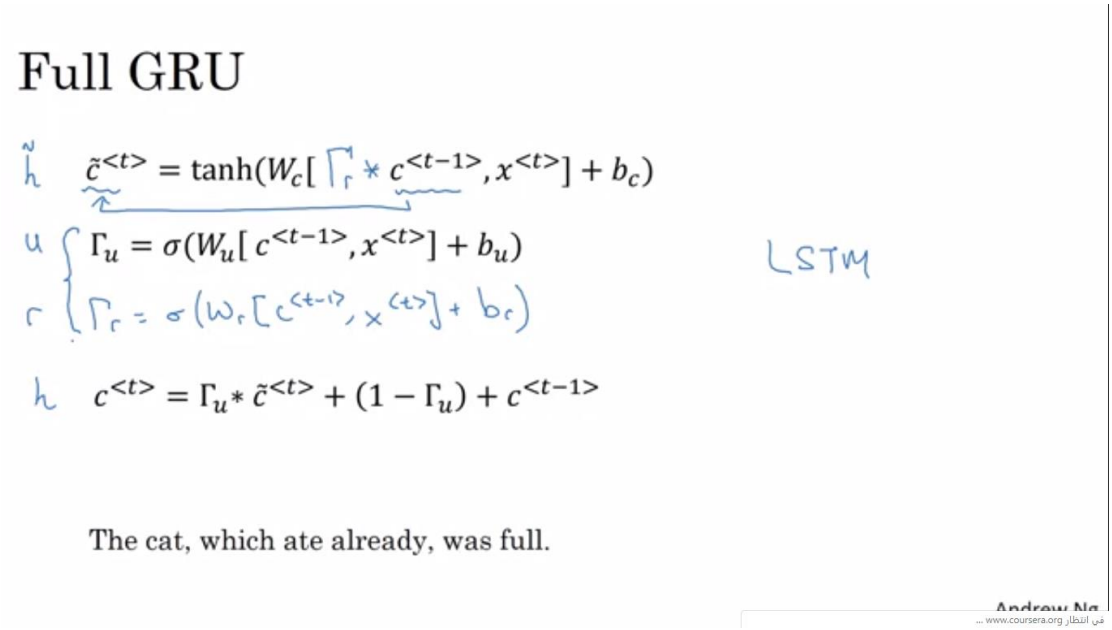
Vanishing gradients with RNNs

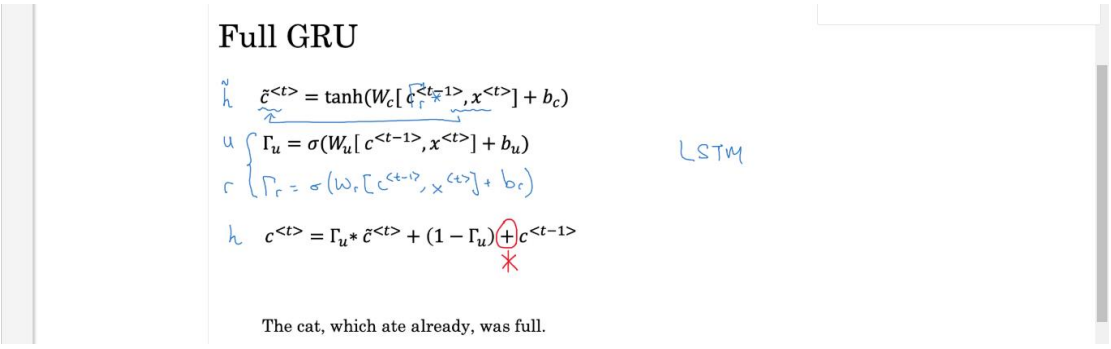


Andrew Ng

To solve this, we use Gated Recurrent Unit (GRU), where we have memory for a word (e.g. Cat) and define long term dependency (e.g was) then update the memory.:







Other more powerful unit than GRU is Long-Short-Term-Memory (LSTM) where we use three gates instead of two in GRU (update, forget and output):

GRU and LSTM

GRU

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

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

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

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

LSTM

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

(update) $\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

(forget) $\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

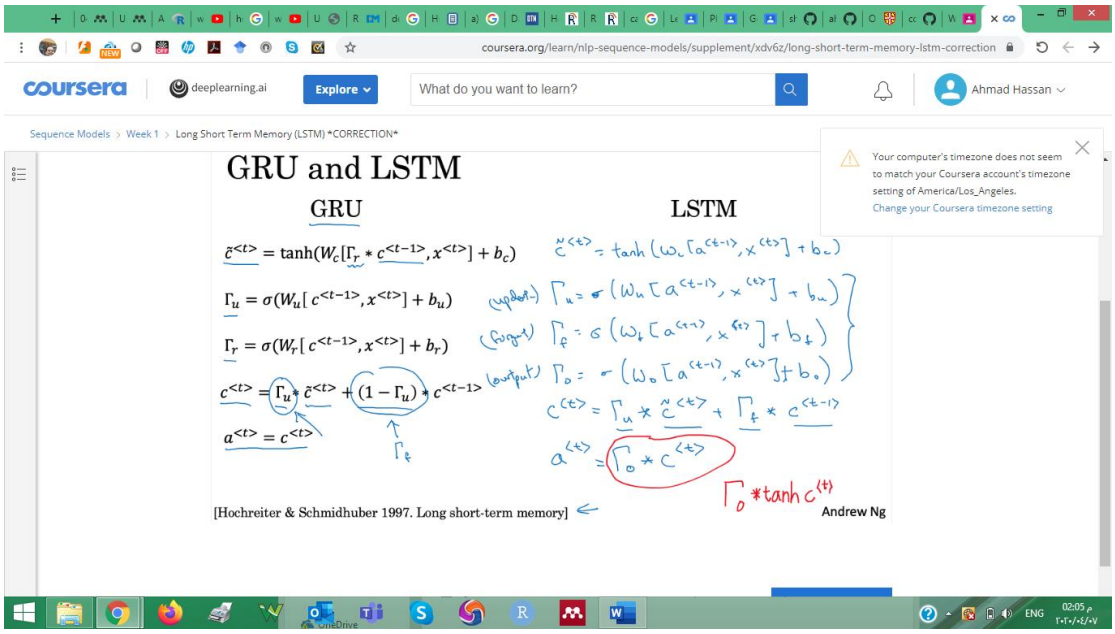
(output) $\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

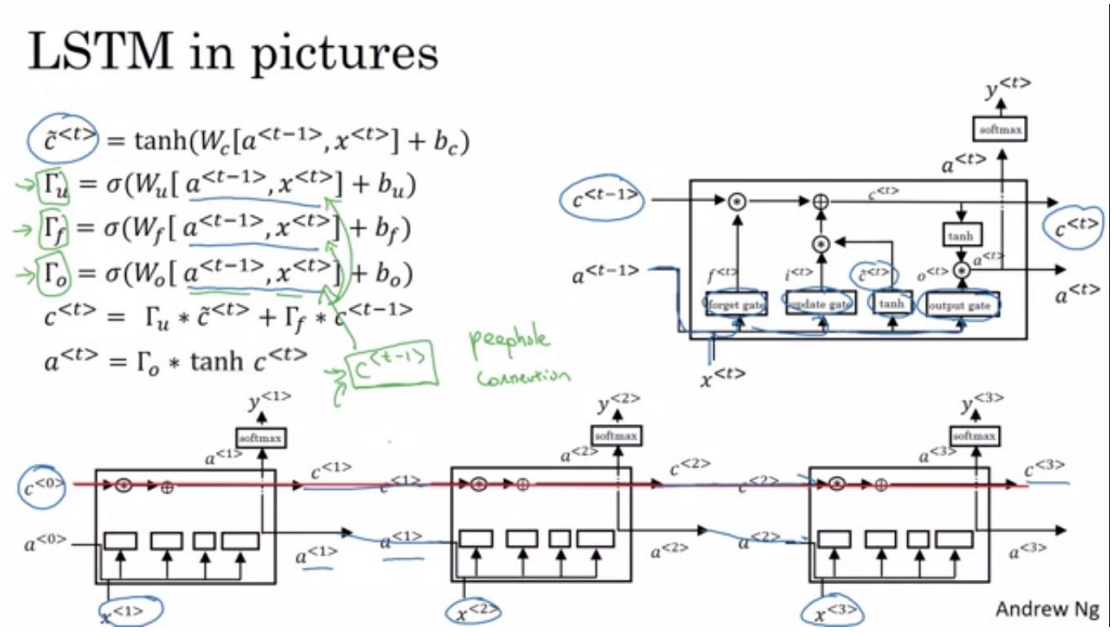
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

[Hochreiter & Schmidhuber 1997. Long short-term memory] ←

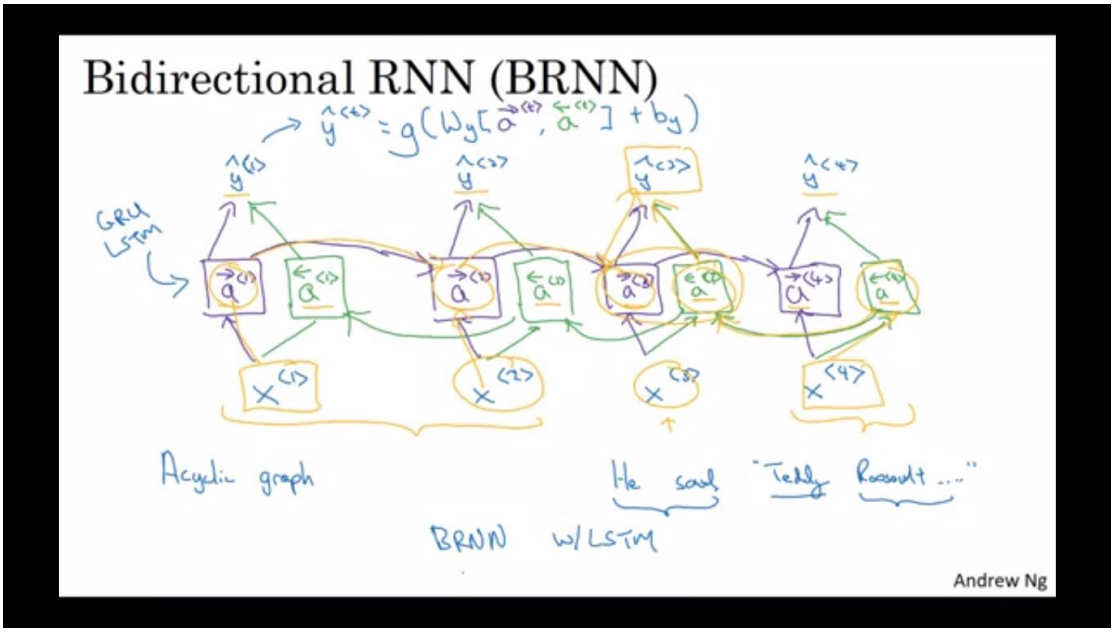
Andrew Ng





Some people use $c^{<t-1>}$ instead of $x^{<t>}$ in gates calculations as they are one-to-one (as not all vector elements affect the specific gate).

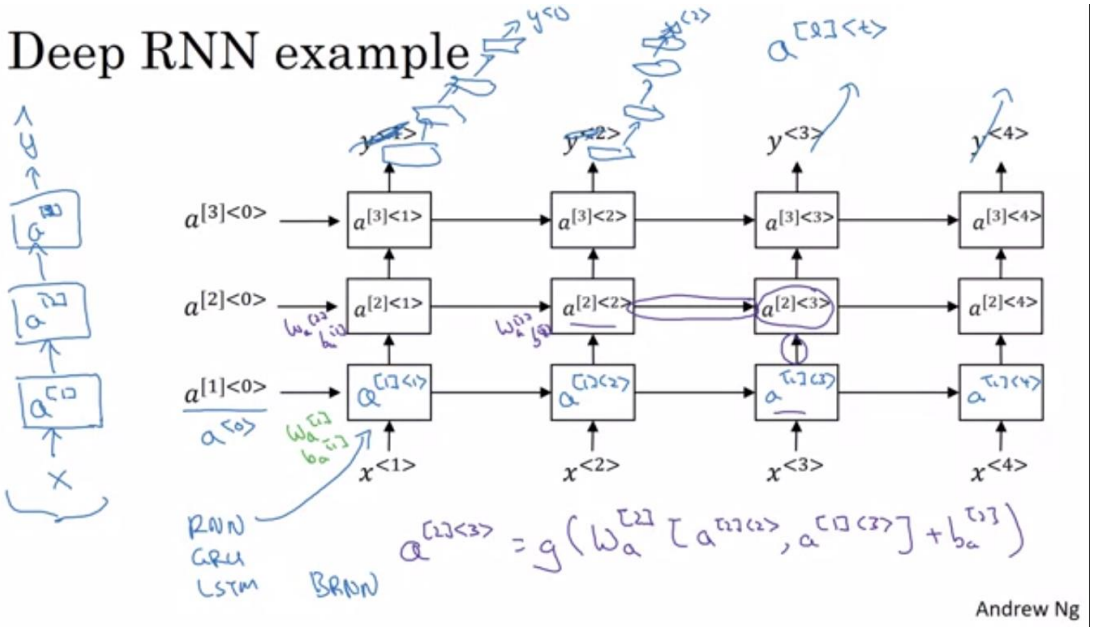
We use Bidirectional RNN (BRNN) to predict the output of single word from inputs from past, present and future. We can use either GRU or LSTM in the BRNN, most common is BRNN with LSTM.:



The BRNN main disadvantage is that it needs the full text already known to work well. So, it works in NLP but bad in speech recognition.

Deep RNN is stacking the any type of previously mentioned RNNs to make Deep NN.

Deep RNN example



Andrew Ng