NLP and Deep Learning
MAT3399

Lecture 5:
Recurrent Neural Networks (RNN)

Tuan Anh Nguyen @ Aimesoft
ted.nguyen95@gmail.com

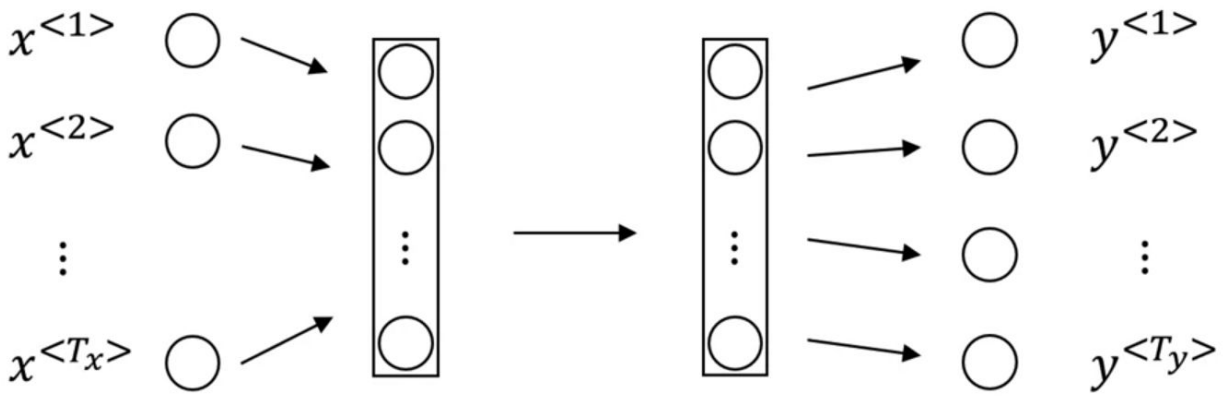
Content taken from Deep Learning Specialization course from Coursera and CS224 from Stanford

Why not use normal network?

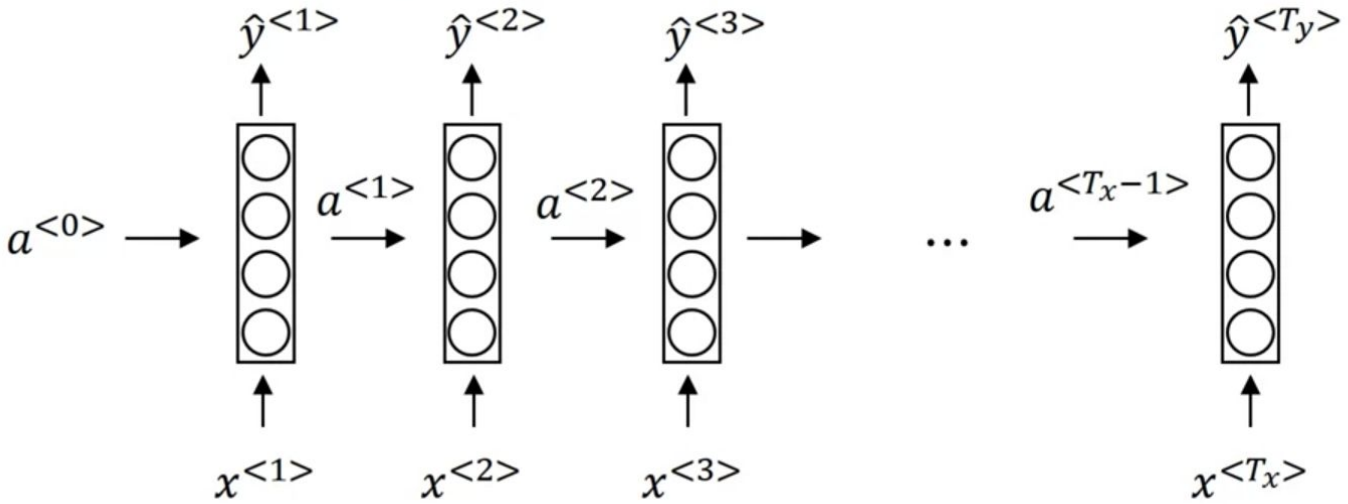
Remember how we use word2vec embeddings in the coding exercise from lecture 3?

Problems:

- Input and outputs could have different lengths
- Missing information



Recurrent Neural Networks (RNN)



Tanh / ReLU

Based on the problem

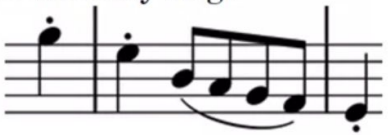
$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

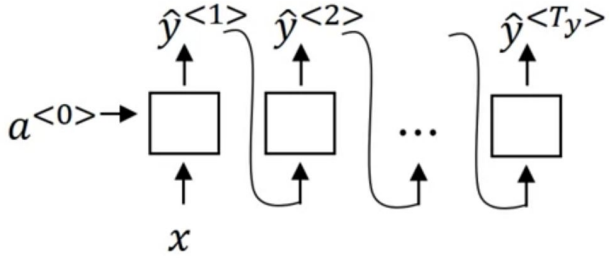
Simplified version

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

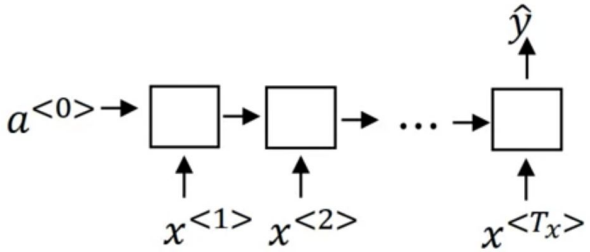
Example of Sequence Data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

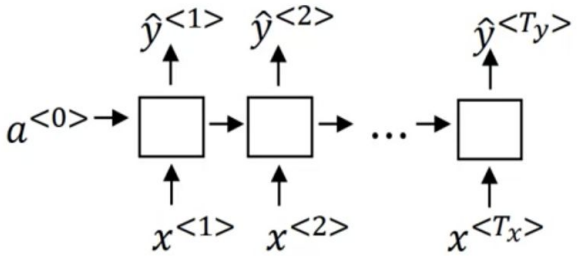
RNN Types



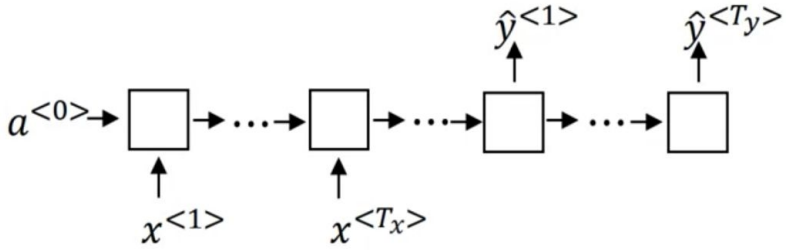
One to many



Many to one



Many to many

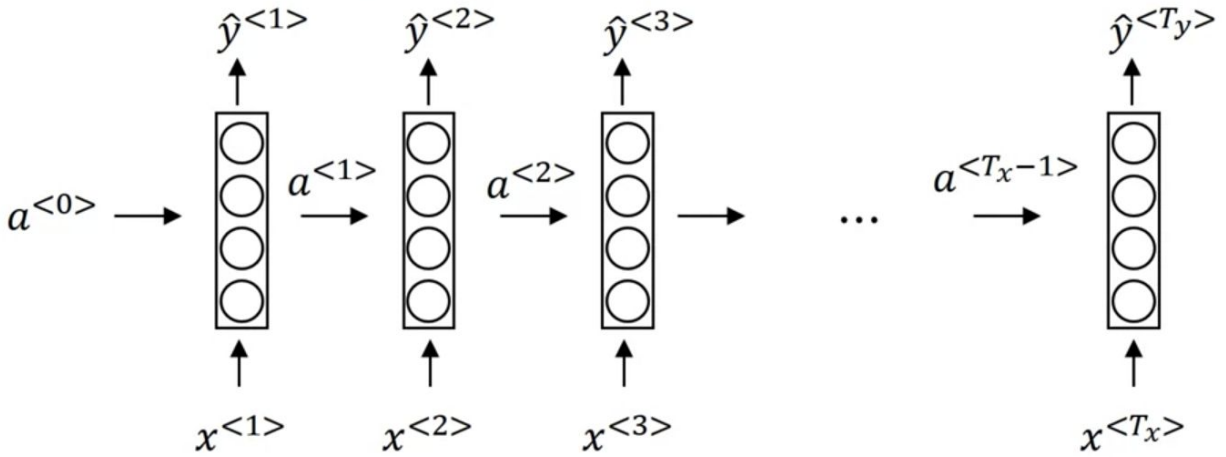


Many to many

Vanishing Gradient Problem in RNN

The cat, which already ate .....was full

The cats, which already ate .....were full



Tx and Ty is very big

RNN tends to forget information that appears early in a very long sequence

Simplified Gated Recurrent Unit (GRU)

Memory cell: c
$c^{<t>} = a^{<t>}$

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

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

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

-> We can carry the information from an early time step to a much later time step

Full GRU

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

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

Relevant / Reset gate

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

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

LSTM

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

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

Forget gate

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

Output gate

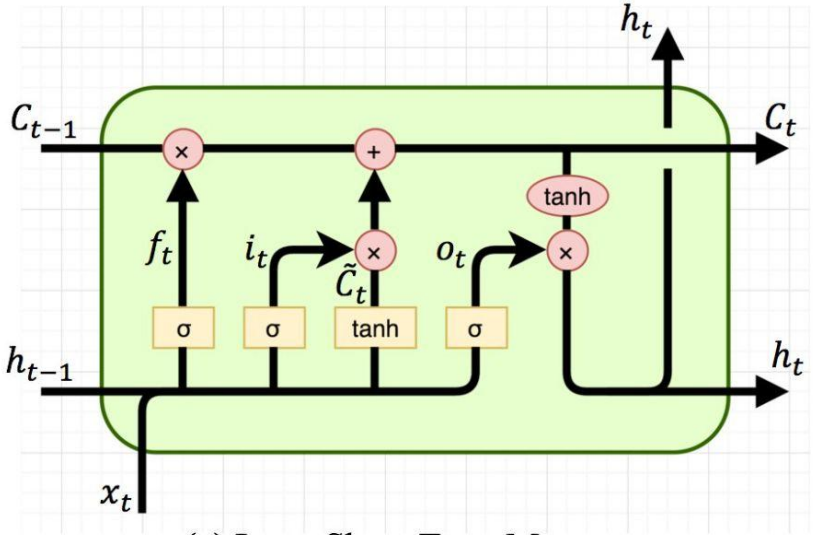
$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

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

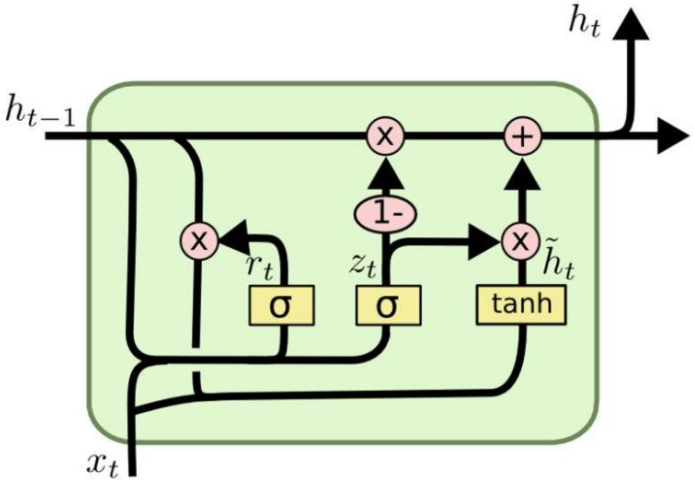
$$a^{<t>} = \Gamma_o * \tanh\, c^{<t>}$$

GRU and LSTM in picture

Note that the notation is slightly different (Because I cannot find any image that has the same notation as previous slides :<)



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

RNNs improved perplexity greatly compared to what came before

n-gram model →

Increasingly complex RNNs ↓

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

↓ Perplexity improves (lower is better)

Generating Text with RNN models

- You can train an RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Harry Potter:

> "Sorry," Harry shouted, panicking—"I'll leave those brooms in London, are they?"
>
> "No idea," said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry's shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn't felt it seemed. He reached the teams too.

Source: https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6

Coding Exercise

- Implement RNN or LSTM for text classification (Use the data from lecture 3)

Advanced exercise: Train a LM using RNN/LSTM