COURSE 5: SEQUENCE MODELS

When the points in the dataset are dependent on the other points in the dataset, the data is called Sequential.

Week 1:

Video 1: Why sequence models?

Applications:

(i) speech recognition (sequence to sequence)
X: input eg. audio (wave sequence)
Y: output eg. text (text sequence)

(ii) music generation
X: ϕ
Y: wave sequence → music

(iii) sentiment classification
X: text
Y: integer rating (stars) (1 to 5)

(iv) DNA sequence analysis
X: DNA sequence
Y: DNA sequence identification etc.

Video 2: **Notation**

Named Entity Recognition Example

x: "Harry Potter and Hermione Granger invented a new spell."
y: 1 1 0 1 1 0 0 0 0

1 - name
0 - not name

same size of i/p & o/p sequence

* n^{th} element of an input sequence is represented as $x^{<n>}$.
example: 1st by $x^{<1>}$ = Harry.
2nd by $x^{<2>}$ = Potter

* n^{th} element of output sequence (y) is represented as $y^{<n>}$.
example: $y^{<1>} = 1$
$y^{<2>} = 1$

* T_x = size of input sequence
T_y : size of output sequence

* $i \rightarrow$ training example

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

$T_x^{(i)}$: input sequence length of i^{th} training example
$T_y^{(i)}$: output sequence length of i^{th} training example

Representing words:

(i) Build a dictionary. It can have 10,000 / 50,000 / 1,00,000 words.

(ii) dictionary = vocabulary list
- each word has unique index representation
- Sorting is in alphabetical order.

(iii) One hot encoding system - digit '1' is used to represent a word in the dictionary at its particular index. Others are 0.

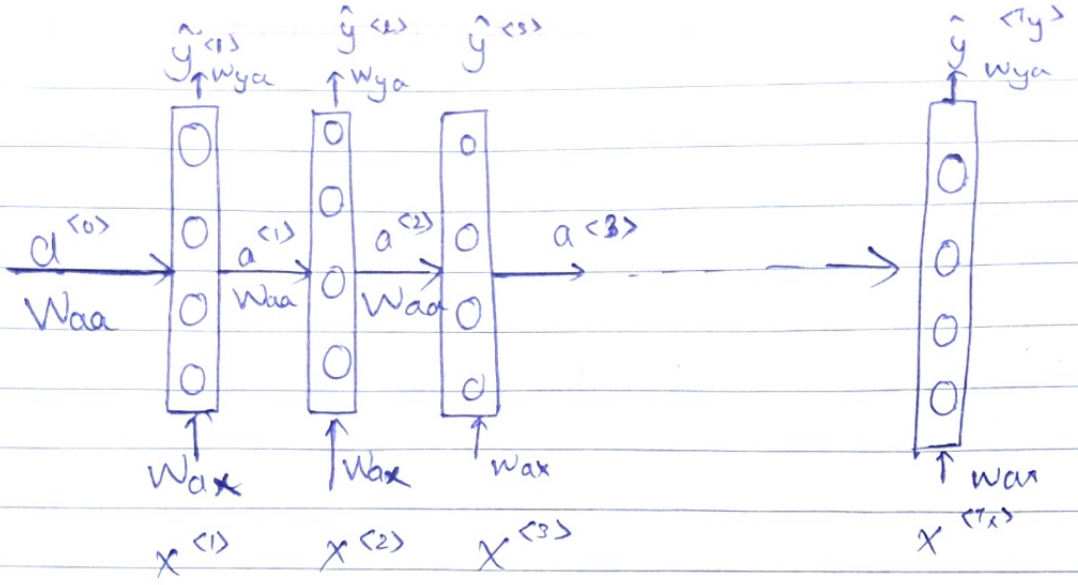
(iv) If there is an unknown word we can add it to our dictionary.

Video 3: **Recurrent Neural Networks**

Problems faced by standard neural networks in the above problem:

- inputs & outputs are of different length
- doesn't share features ~~across~~ learned across different positions of text.

$T_x = T_y$



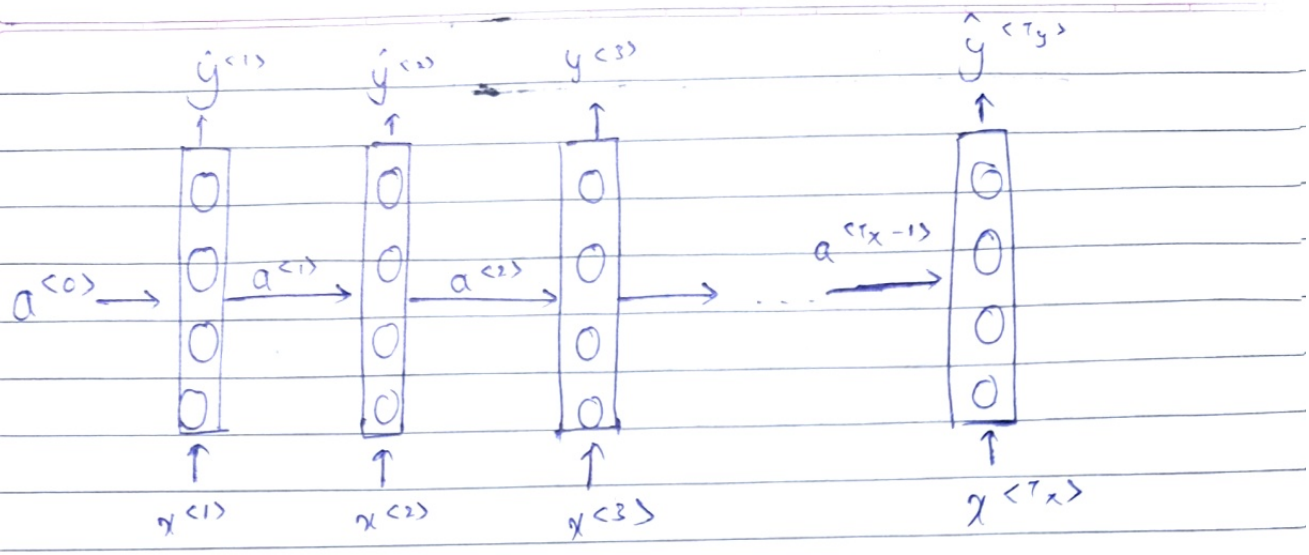
$a^{<0>}$ - it is initialised with zeros

Wax : (No. of hidden neurons, n_x)
Waa : (No. of hidden neurons, No. of hidden neurons)
Wya : (n_y, no. of hidden neurons)
L → R

Wax - parameters used in each step
Waa - activation parameters
Wya - governs output

$\hat{y}^{<t>}$ depends on previous input and activations

RNN's use information that is earlier in the sequence to predict. They cannot access tokens after the word in the sequence.



$a^{<0>} = \vec{0}$

$a^{<1>} = g_1(W_{aa}\, a^{<0>} + W_{ax}\, x^{<1>} + b_a)$ ← tanh / ReLU

$\hat{y}^{<1>} = g_2(W_{ya}\, a^{<1>} + b_y)$ ← sigmoid

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

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

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

W_{aa} : $(100, 100)$

$a^{<t>} = g(\underset{(100,100)}{W_{aa}}\, a^{<t-1>} + \underset{(100,10000)}{W_{ax}}\, x^{<t>} + b_a)$

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

$W_a = [W_{aa} : W_{ax}]$ ← 100×10000

$(100, 10100)$

$a^{<t-1>}, x^{<t>} = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}$ (100 ↑, 10000 ↓) → 10100

Page No.
Date

Video 4: Backpropagation Through Time

$$\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1 - y^{<t>}) \log (1 - \hat{y}^{<t>})$$

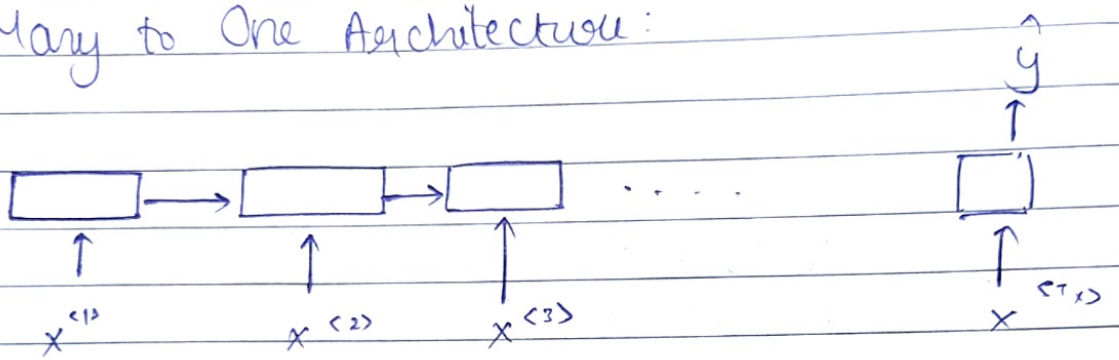
$$\mathcal{L}(\hat{y}, y) = \sum \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

Video 5: **Different types of RNN**

(i) Many to Many Architecture : Input sequence has many inputs as sequence and the outputs sequence also has many outputs.

eg. language translation

(ii) Many to One Architecture:

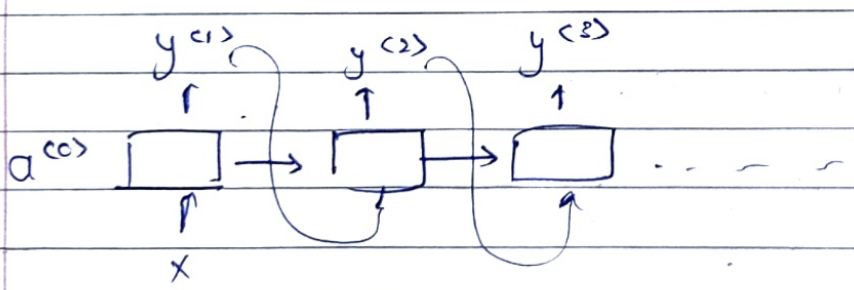


eg. sentiment analysis in movie rating.

(iii) One to many Architecture

X : input

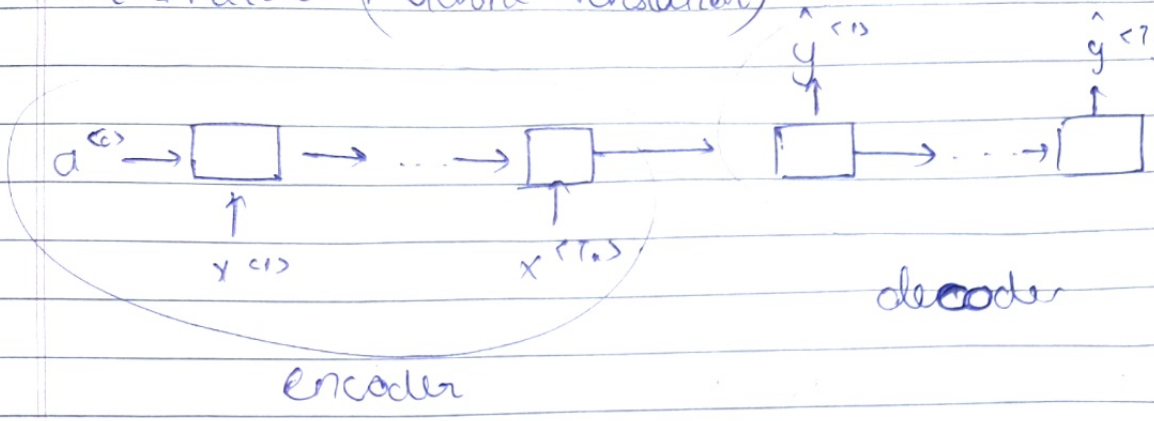
X = ϕ (integer like "genre"



eg. music generation

Page No.
Date

Alternative (Machine Translation)



Video 6: Language Model & Sequence Generation.

* Language model gives the probability of a given sequence of words.
Example:

The apple & pair salad
The apple & pear salad

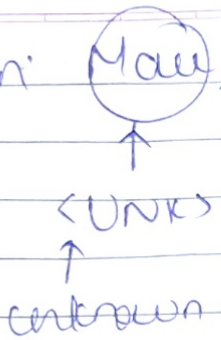
P(The apple & pair salad) = 3.2×10^{-13}
P(The apple & pear salad) = 5.7×10^{-10}

* To build Language Model:

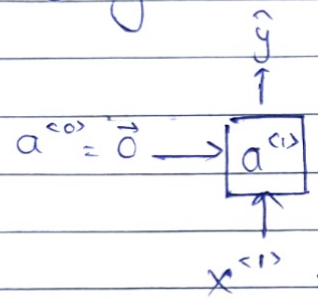
(i) Training set: large corpus (huge body) of english text

(ii) Tokenise
(iii) One hot vectors
(iv) Extra token <EOS> for end of sentence
* you can tokenise punctuation too by adding to vocabulary.

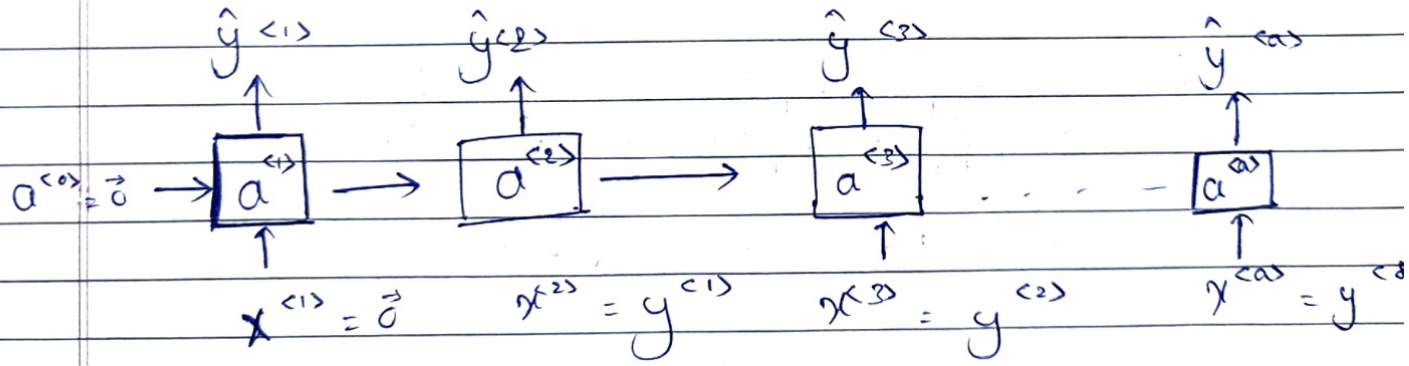
* The Egyptian Mau is a breed of cat. <EOS>



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

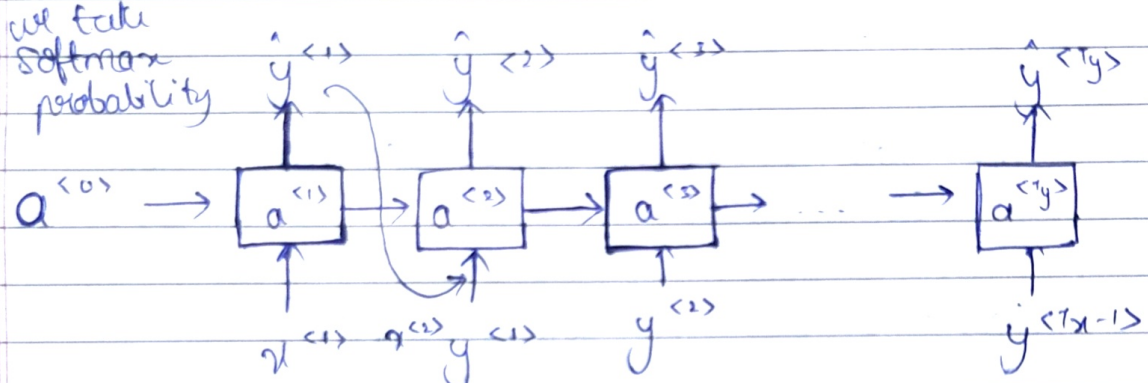


$a^{<1>}$ - makes softmax prediction about probability of first word y



$$P(y^{<1>}, y^{<2>}, y^{<3>}) \leftarrow$$
$$P(y^{<1>}) \, P(y^{<2>} \mid y^{<1>})$$
$$P(y^{<3>} \mid y^{<1>}, y^{<2>})$$

Video7: Sampling Novel Sequences



$a^{<0>}$ — zeros vector
$x^{<1>}$ — zeros vector

Prediction is chosen from distribution obtained by $\hat{y}^{<1>}$. eg. 'The'
In numpy - np.random.choice

Continue till you get <EOS> token
you can reject <UNK> token

* we have to build a word level language model.
* For character level language model -
  vocabulary : [a-z A-Z 0-9], punctuation, special characters / token.

In character level model there will be no <UNK> token.
However you end up with longer sequences. Unlike word level language models they cannot capture long range dependencies between how words in earlier parts effect words in later part. They are harder to train

Video: **Vanishing Gradients With RNN's**

Example:

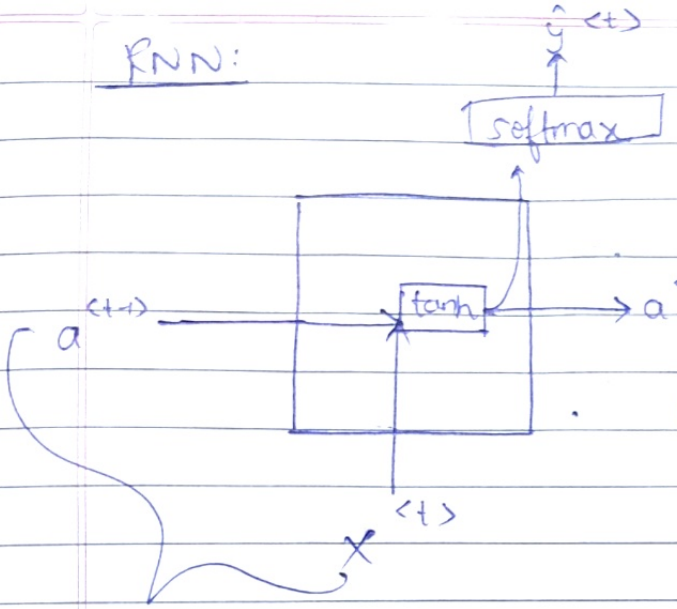
The cat, which already ate ..... , was full.
The cats, which ..... , were full.

cat – was
cats – were
} RNN can't
↓
i.e vanishing gradient

For computing the word "was" we need to compute the gradient for everything behind. Multiplying fractions tends to vanish the gradient while multiplication of large numbers causes it to explode.

Exploding gradients can be seen when weight values become NaN
Solution – gradient clipping
rescale gradient so that it isn't too big

RNN:



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

g → tanh

C = memory cell

$C^{<t>} = a^{<t>}$ at time t

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

Γ_u = value between 0 & 1

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

↑ update gate

C is updated using $\tilde{C}$

gate decides whether we update or not

C is set to 0/1

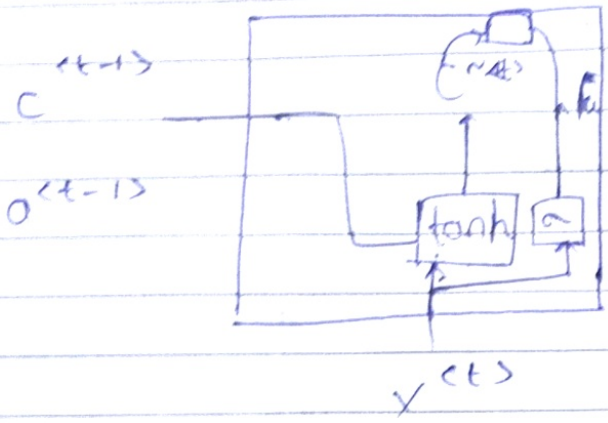
$$C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (1-\Gamma_u) * C^{<t-1>}$$

Example:

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

<1> <1>

GRU:



$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

relevance gate $\Gamma_r = \sigma\left(W_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$

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

Shape of $a^{\langle t \rangle} = c^{\langle t \rangle} = \tilde{c}^{\langle t \rangle} = u^{\langle t \rangle}$

Video 9: Long Short Term Memory (LSTM)

GRU:

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

$$\Gamma_r = \sigma\left(W_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_r\right)$$

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

$$a^{\langle t \rangle} = c^{\langle t \rangle}$$

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

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

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

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

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

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

$c^{<t-1>}$ — Peephole connection

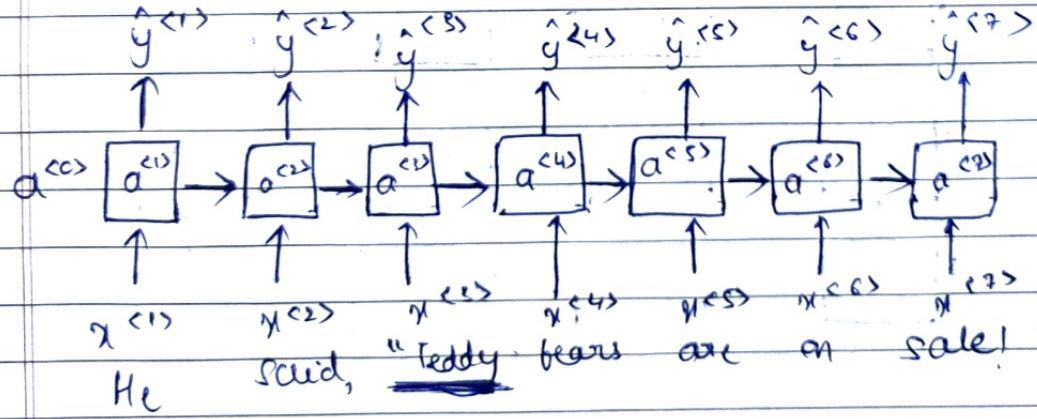
LSTM >>> GRU

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

LSTM & GRU's are good at memorising values. Some real values stored in the memory cell can be used for many many time steps.

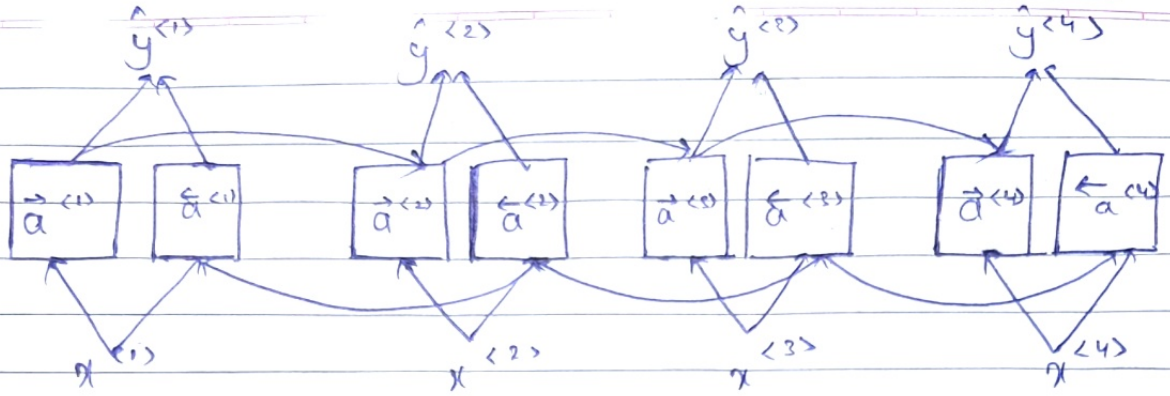
Video 10: **Bidirectional RNN** (BRNN)

* Bidirectional RNN helps you take information from both earlier & later in sequence



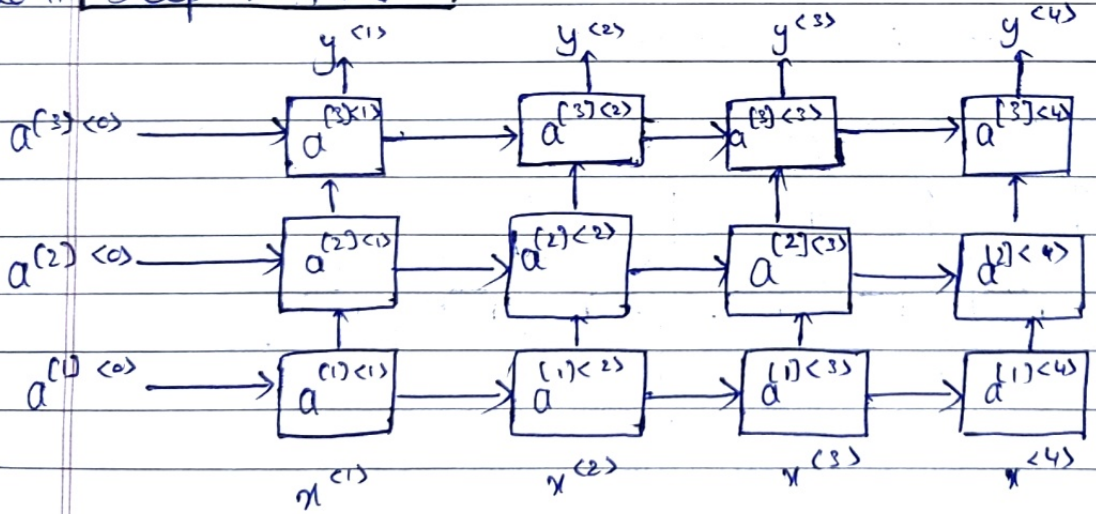
We know what Teddy is based on (bears) and not Helsan

BRNN fixes this issue.



* BiRNN is an acyclic graph
* Part of the forward propagation goes from left to right & part from right to ~~right~~ left.
* Blocks can be RNN/LSTM/GRU
* Disadvantage: whole sequence is required before processing it.

Video 11: **Deep RNN's**



In RNN's layers is considered deep.