C5w1

sequence data :-

Example :-

- Speech recognition
- Music generation
- Sentiment classification
- DNA sequence analysis
- Machine translation
- Video activity recognition
- Name entity recognition.

Notation :-

Let say we want to know the position of person name in a sentence.

X :-	Harry	porter	and	Harmony	Granger	invented	a	new	spell.
	$x^{<1>}$	$x^{<2>}$	$x^{<3>}$	. . .	. . .	. . .	. . .	. . .	$x^{<9>}$
Y :-	1	1	0	1	1	0	0	0	0
	$y^{<1>}$	$y^{<2>}$	$y^{<3>}$	. . .	. . .	. . .	. . .	. . .	$y^{<9>}$

Each of the word is represented by $x^{<i>}$ and their output $y^{<i>}$

- Numbers of word → $T_x = 9$
  $T_y = 9$

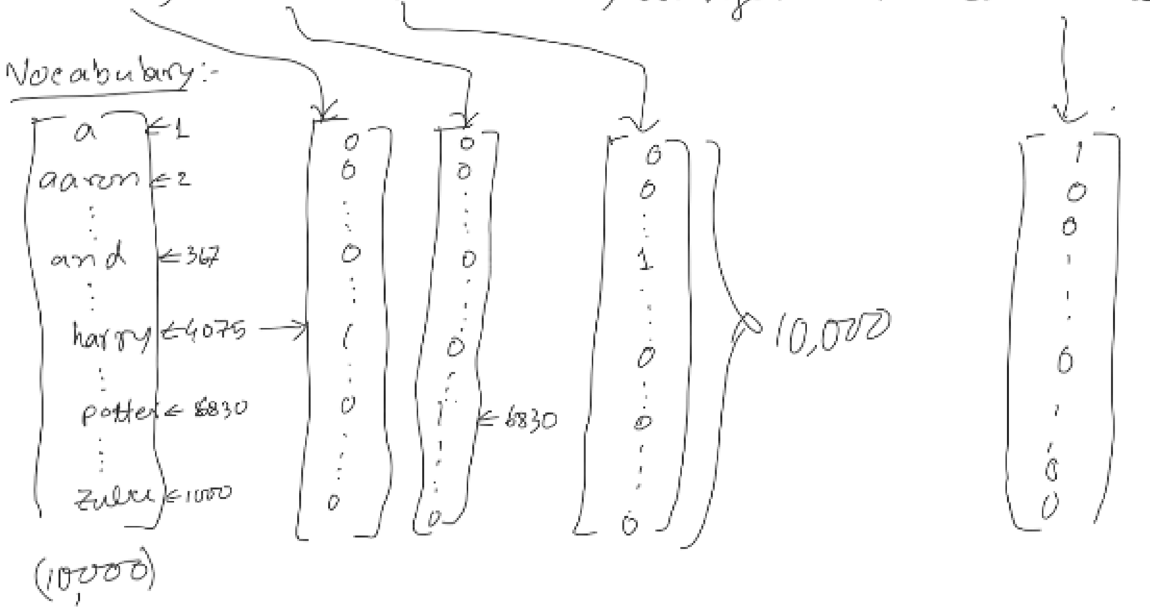
$x^{(i)<t>}$ → ith training example t-th word.

$T_x^{(i)}$ → length of ith training example.

Representing words:-

We maintain a vocabulary of words & use a one hot vector to represent each words of a sentence. One-hot means there will be one "1" in the vector & rest are zero. And that "1" will be in the same position of the word in the vocabulary.

X:- Harry potter and Harmony Gringer invented a new spell.



This single line will be a (10000, 9) matrix.
If a word doesn't present in the vocabulary then all those word will contain one position, which will refer to all unknown words.

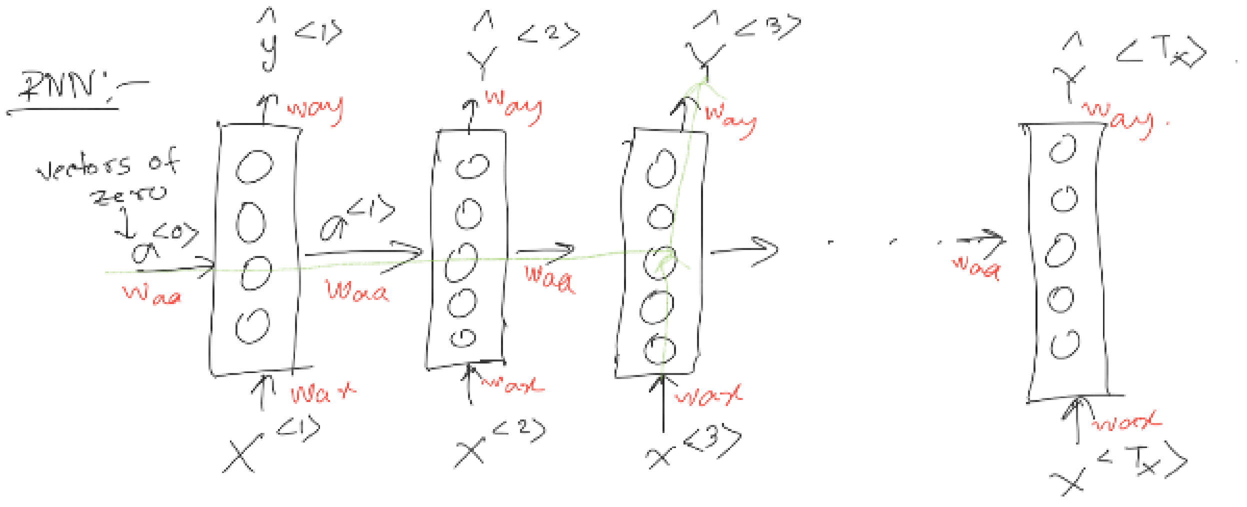
RNN (Recurrent neural network):-

why not standard network:-

- Input, outputs can be different lengths in different example.
- Doesn't share features learned accross different position of text.

- Inputs are pretty big.

RNN:-



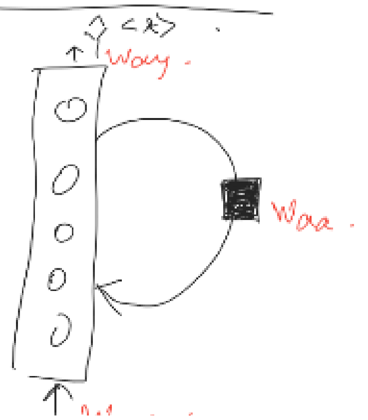
if we want to predict $\hat{y}^{<3>}$ we use the previous data as well ($x^{<1>}$, $x^{<2>}$) those informations are passed through recurrent network. But we cannot use latter information ($x^{<4>}$, $x^{<5>}$...). And it's a biggest disadvantages of RNN

For example:- we want to extract the name from following two sentences.

- He said, "Teddy roosevelt was a great president"
- He said "Teddy bears are on sale".

RNN have only idea about previous word (He said). Both the teddy in the two sentences will give same result. But those are different. If RNN have the idea about later information it could have guess the word correctly.

Other interpretation:-



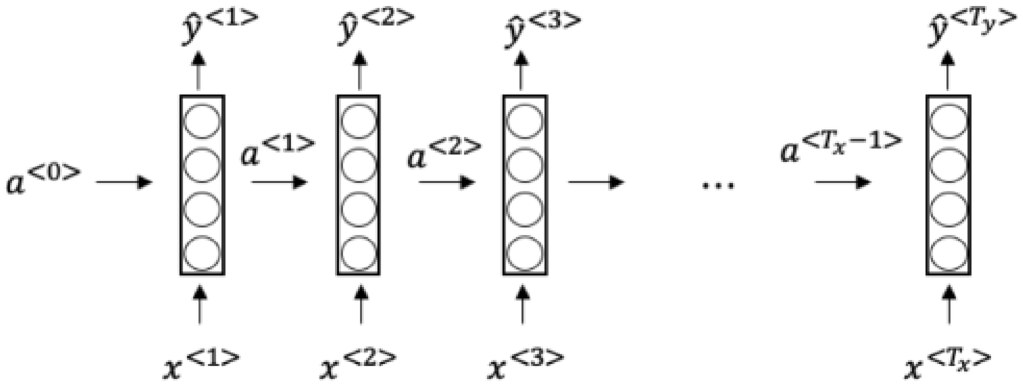
*Loop denote their recurrent connection

*shadded box denote the time delay of one steps.

W_{ax}

$x^{<t>}$

Forward propagation:-



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

$a^{<1>} = g(W_{aa}a^{<0>} + W_{ax}x^{<1>} + b_a)$ ← tanh/Relu. ← more common

$\hat{y}^{<1>} = g(W_{ya}a^{<1>} + b_y)$ ← sigmoid/softmax.

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

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

Notation:-

W_{ax} → it will be multiplied with x like quantities to compute a like quantities.

simplified RNN notation:-

$$a^{<t>} = g(\underbrace{W_{aa}}_{(100,100)}\underbrace{a^{<t-1>}}_{100} + \underbrace{W_{ax}}_{(100,10000)}\underbrace{x^{<t>}}_{10,000} + b_a)$$

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

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

$$= g(W_y a^{<t>} + b_y)$$

$$\underset{(100,10100)}{W_a} = \underbrace{\left[\underbrace{W_{aa}}_{100} \vdots \underbrace{W_{ax}}_{10,000}\right]}\updownarrow 100$$

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ \hline x^{<t>} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 10000 \end{matrix} \updownarrow 10,100$$

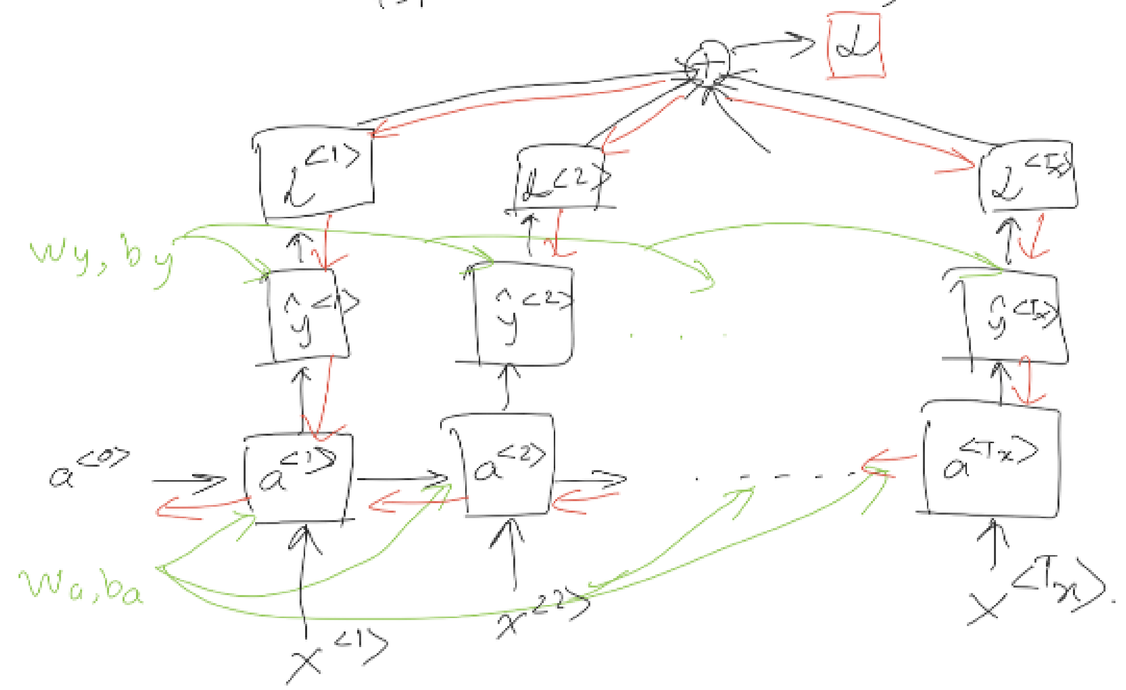
$$[W_{aa} \vdots W_{ax}] \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} =$$

$W_{aa}a^{<t-1>} + W_{ax}x^{<t>}$.

Backpropagation:-

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

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



This called backpropagation through time.

Different types of RNN:-

The examples we have seen so far is :- $T_x = T_y$.

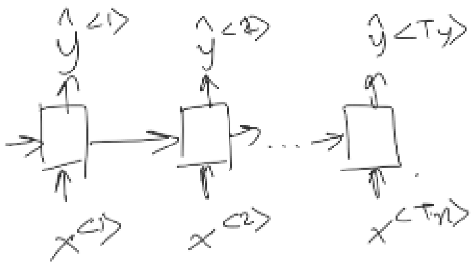
But there are many cases where $T_x \neq T_y$. For example

machene translation:- translated word might not have the same number of word as original sentence.

sentiment analysis :- Input can have multiple word but output might have integer value.

| Sentiment classification |

$T_x = T_y$

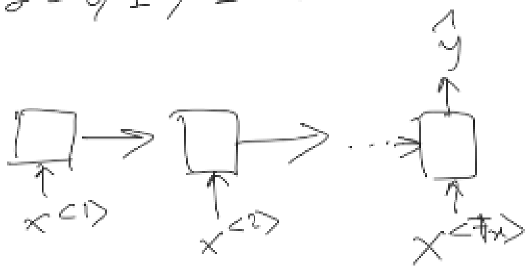


Many-to-many

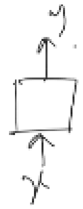
Sentiment classification:-

x = text

$y = 0/1 \;/\; 1 \ldots 5$



many-to-one

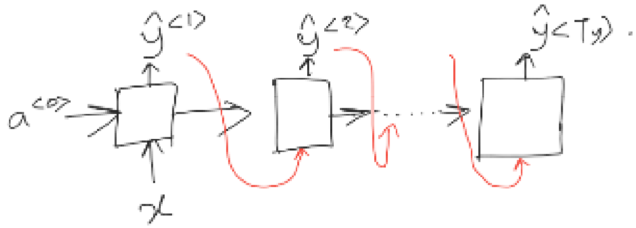


one-to-one

Music generation.

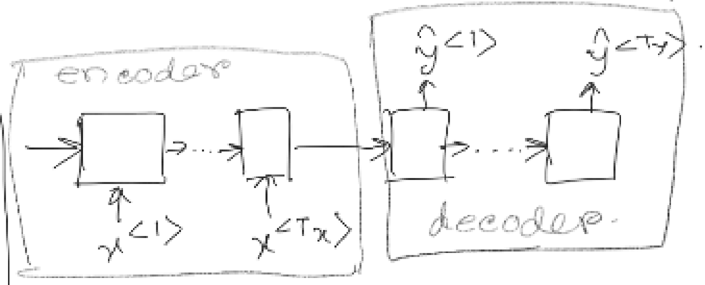
can be music genre

$x \rightarrow \hat{y}^{<1>} \hat{y}^{<2>} \ldots \hat{y}^{<T_x>}$.



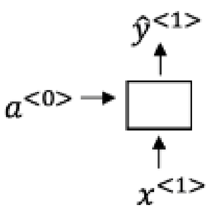
one-to-many

$T_x \neq T_y$ (e.g. Machine translation)

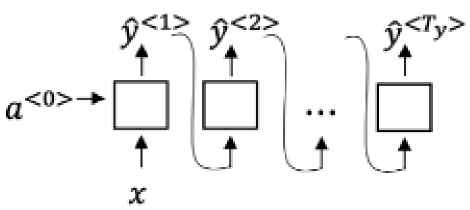


Many-to-many

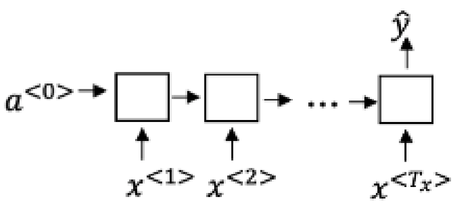
Summary:-



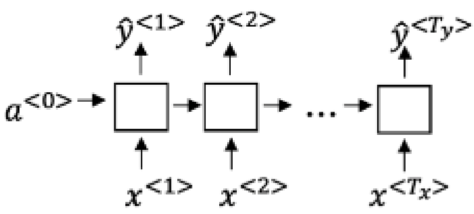
One to one



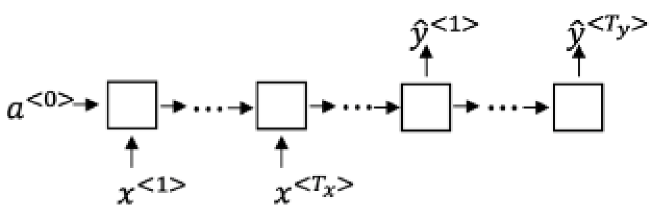
One to many



Many to one



Many to many



Many to many

Language modeling:-

While we speak some of the word might have the same pronounciation. Language modeling provides the probability of each of the sentence that have similar pronouncing word.

For example:-

$$P(\text{Apple of pair salad}) = 3.2 \times 10^{-13}$$

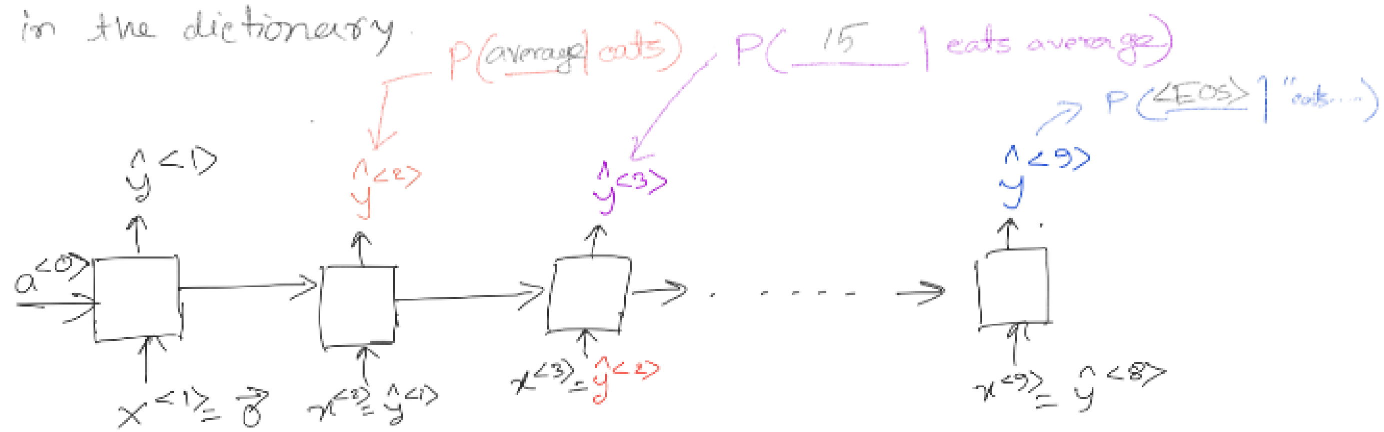
$$P(\text{Apple of pear salad}) = 5.7 \times 10^{-10}$$

RNN model:-

End of sentence

Cats average 15 hours of sleep a day <EOS>

softmax, that will predict the probability of each word happening in the dictionary.



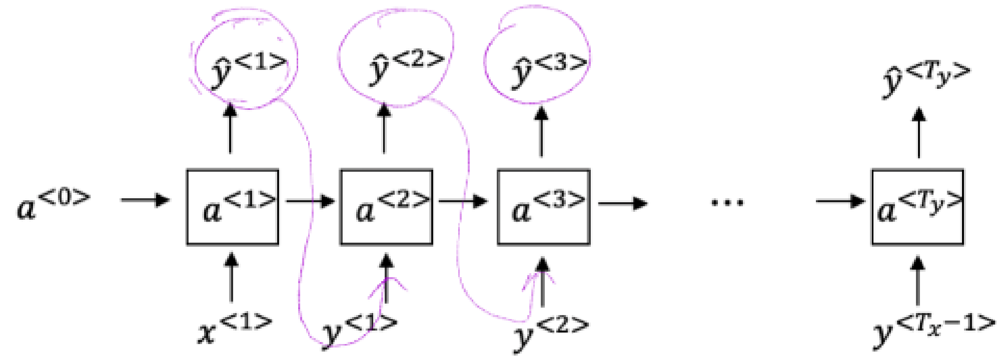
Cost function:-

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

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

Sampling a sequence from a trained RNN:-

this will provide a softmax output. Insted of using the single word from softmax. We will use a random word based on the probability that softmax provided.



Character level language model:-

Instead of using word in the vocabulary, we will use character in the vocabulary. So each RNN block will compute a single character instead of the a word.

Vanishing gradient with RNN:-

The RNN that we have seen so far are not super good to capture long term dependency in a sentence. for example.

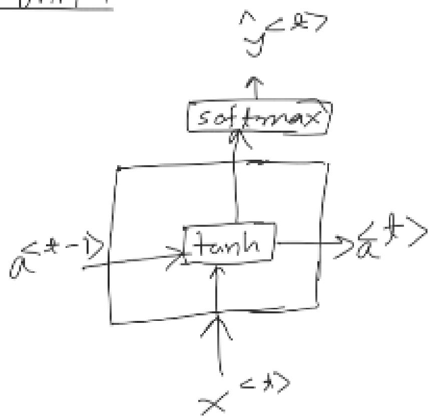
The cat, which already ate ......, was full.
The cats, which already ate ......, were full.

Exploding gradient is easier to solve. If the gradient surpass a threshold value then we can normalize the gradients this is also called gradient clipping

Gate Recurrent Unit (GRU):-

- capture long range connection.
- Mitigate the vanishing gradient problem.

RNN unit:-



tanh.

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

GRU simplified

c = memory cell.

$\rightarrow c^{<t>} = a^{<t>}$.

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

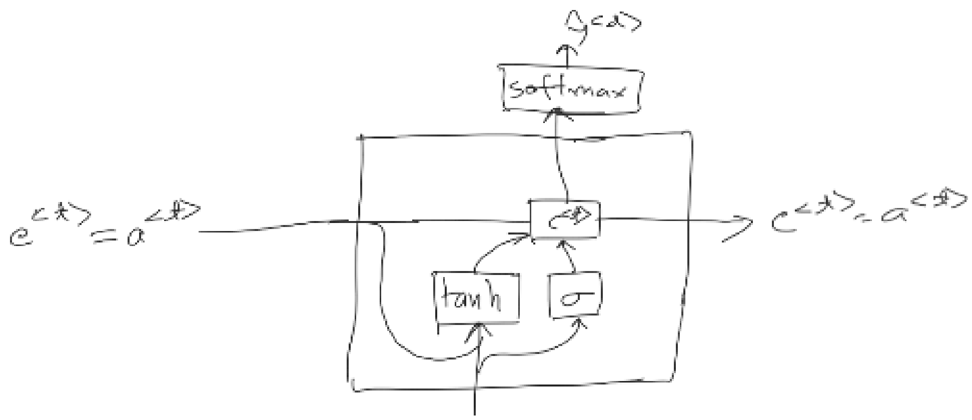
$\rightarrow \Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u]$

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$ | update the $c^{<t>}$ if Γ_u becomes 1. Otherwise stick with the previous one.

$\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ . . . . . $c^{<t>}$ remembers if the cat was singular or not.

$c^{<t>} = 1$ $c^{<t>} = 1$ $c^{<t>} = 1$ . . . . $= 1$ if $c^{<t>} = 1$ it's singular.

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



$x^{<t>}$

Full GRU:-

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

How relevant $c^{<t-1>}$ to compute next candidate $c^{<t>}$.

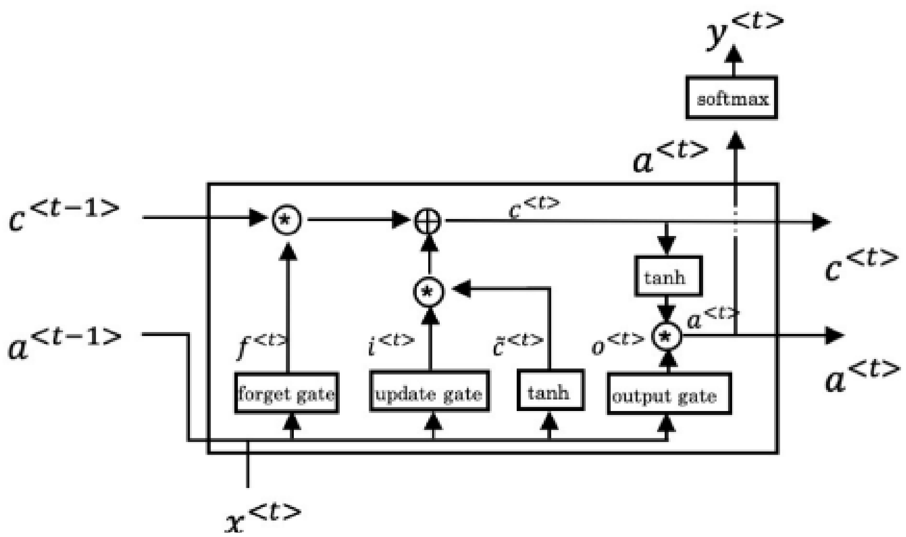
$$\Gamma_u = \sigma(W_u[\,c^{<t-1>}, x^{<t>}] + b_u)$$

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

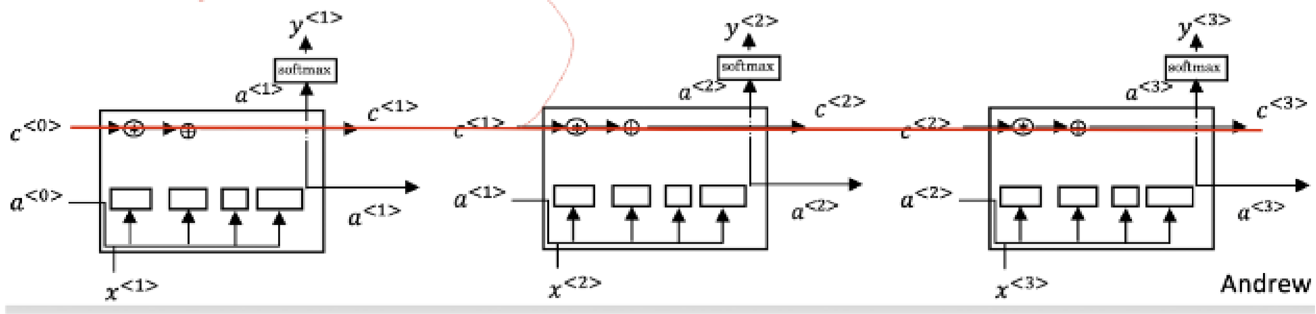
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

Long-short term memory:-

GRU	LSTM
$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$	$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$
$\Gamma_u = \sigma(W_u[\,c^{<t-1>}, x^{<t>}] + b_u)$	$\Gamma_u = \sigma(W_u[\,a^{<t-1>}, x^{<t>}] + b_u)$
$\Gamma_r = \sigma(W_r[\,c^{<t-1>}, x^{<t>}] + b_r)$	forget $\Gamma_f = \sigma(W_f[\,a^{<t-1>}, x^{<t>}] + b_f)$
$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$	output $\Gamma_o = \sigma(W_o[\,a^{<t-1>}, x^{<t>}] + b_o)$
$a^{<t>} = c^{<t>}$	$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$
	$a^{<t>} = \Gamma_o * \cancel{c^{<t>}}\ \tanh(c^{<t>})$



This line enables LSTM to remember certain value even for many many time steps.



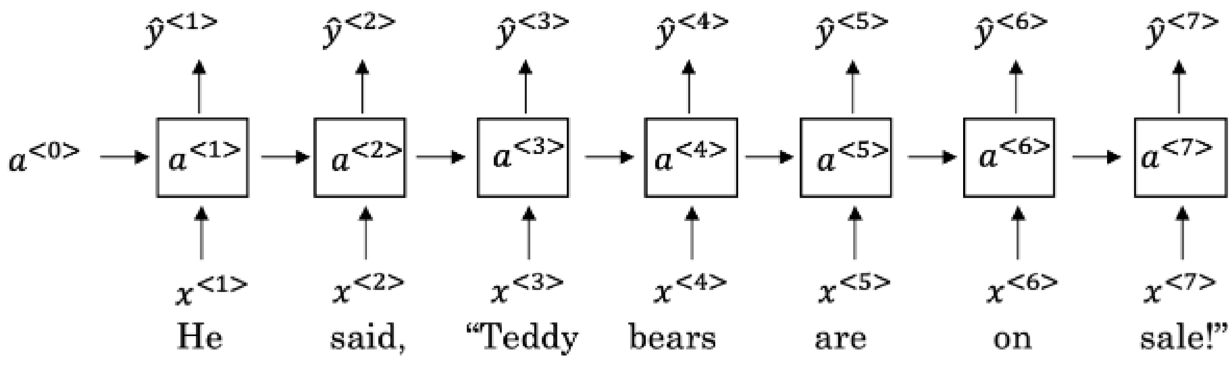
Bidirectional RNN (BRNN):-

Problem with unidirectional:

This is a name entity problem. that extract name from sentence. Unidirectional RNN doesn't have enough information to classify teddy as name.

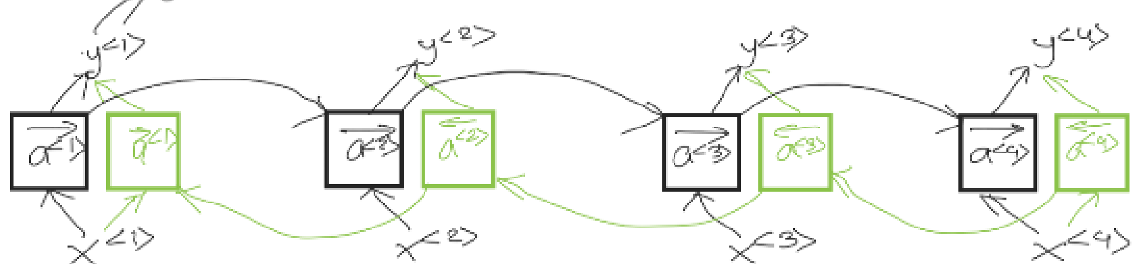
He said, "Teddy bears are on sale!"

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



BNN:-

$$\hat{y}^{<t>} = g(W_y[\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

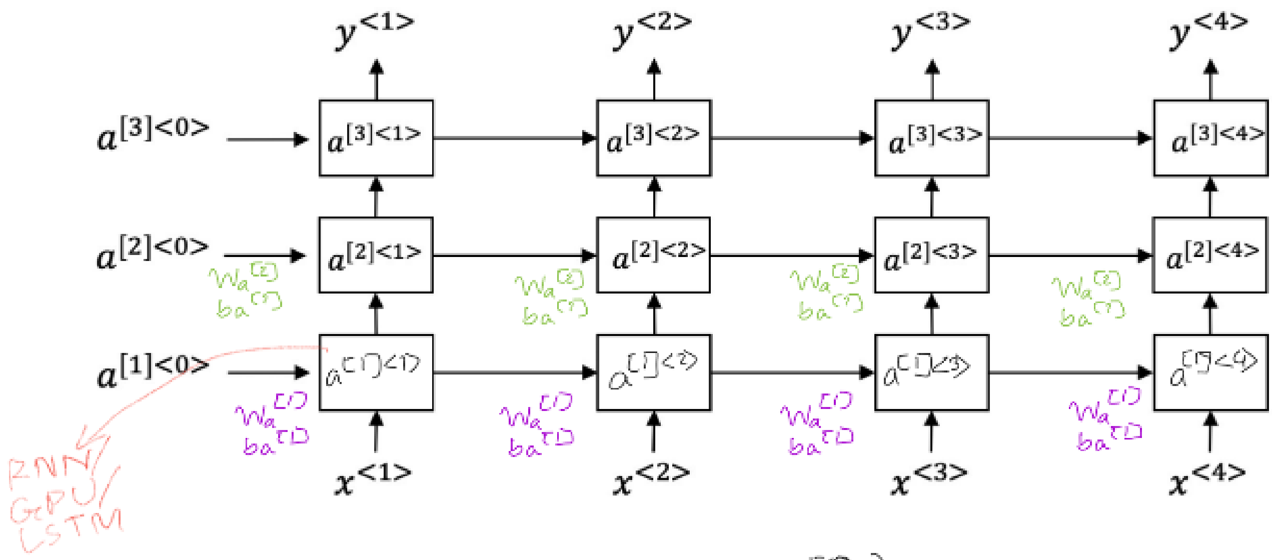


→ If we build a speech recognition system using BRNN then we have to wait untill a person stops talking.

Deep RNN:-

$a^{[l]<t>}$

| l = layer
| t = timestamp



$$a^{[2]<3>} = g\left(W_a^{[2]}\left[a^{[2]<2>}, a^{[1]<3>}\right] + b_a^{[2]}\right)$$

every layer will have their own weights. which will be used in each of the timestamp.