# Course 5: Sequence Models

## Week 1: RNNs

1) Why RNNs (Supervised learning: X → Y)
(as training set)

eg) Speech recognition

Audio X → Text transcript Y
sequence data (both i/p & o/p)

eg) Music generation

eg) Sentiment classification

→ "there is nothing to like in this movie" → 1★/5★ (rating)

eg) DNA sequence analysis

eg) Machine translation

eg) Video activity recognition

"clip of a race of usain bolt" → Running

eg) Name entity recognition

Yesterday, Harry Potter met Hermione Granger → Yesterday, Harry Potter met Hermione Granger

Page No.
Date

2) Notation :

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<t>}$ ...

x : Harry Potter and Hermione Granger invented a new spell. $x^{<9>}$

$T_x = 9$

y : 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<9>}$

$T_y = 9$

$x^{(i)<t>}$

$T_x^{(i)}$ → input sequence length for training example (i).

$y^{(i)<t>}$ → tth element in the output sequent for the ith example

$T_y^{(i)}$ → length of output sequence of ith example

## Representing words (One-hot representation)

x : $x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<9>}$

Vocabulary

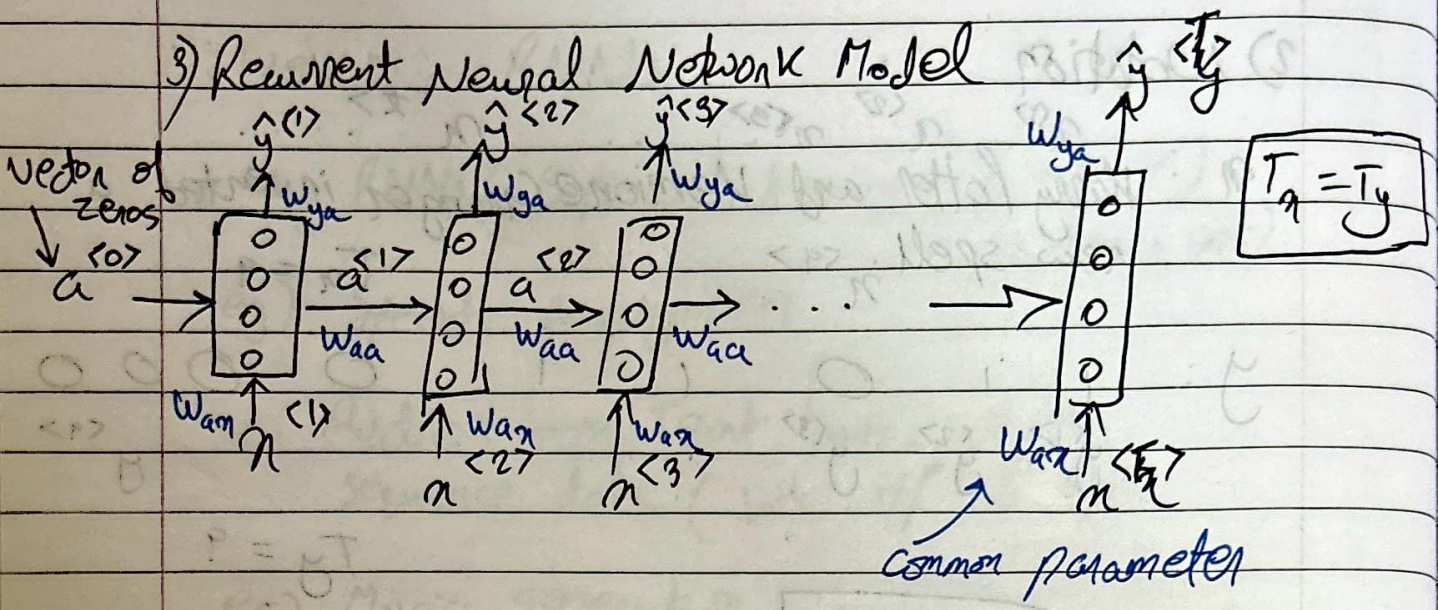
| | |
|---|---|
| a | 1 |
| and | 367 |
| harry | 4075 |
| potter | 6830 |
| ⋮ | 10,000 |

$x^{<1>}$ → 4075 ; $x^{<2>}$ → 6830 ; $x^{<3>}$ → 367

dim = (10000, 1)

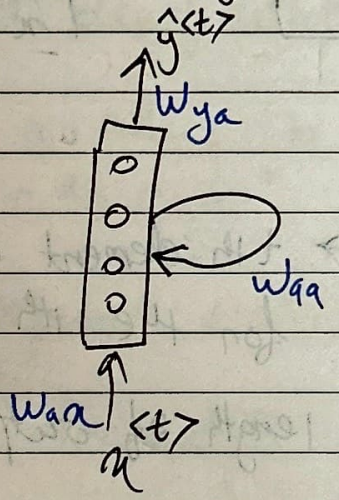
Page No.
Date

## 3) Recurrent Neural Network Model




* RNN scans data from left to right

Rolled diag:




Drawback: while predicting $\hat{y}^{<t>}$, the output has only access to its previous inputs, so this can lead to ~~this~~ some problems and hence inconnect outputs. for ex:

ex1) He said, "Teddy Roosevelt was a great president."

ex2) He said, "Teddy Bears are on sale!"

Now just by knowing the first 3 words of the

$a \leftarrow w_{ax} \cdot x^{\langle 1 \rangle}$

used to compute a like quant. ↑ Mult. by x like quantity

Page No.

above sentences we cannot classify Teddy as a name of a person or not.

This issue can be resolved using BRNNs i.e. Bidirectional RNNs.

Forward propagation of RNN:

$$a^{\langle 0 \rangle} = \vec{0}$$

$$a^{\langle 1 \rangle} = g_1(w_{aa} a^{\langle 0 \rangle} + w_{ax} x^{\langle 1 \rangle} + b_a)$$

$$\hat{y}^{\langle 1 \rangle} = g_2(w_{ya} a^{\langle 1 \rangle} + b_y)$$

(maybe some other activation fn)

$g_1 \rightarrow$ tanh / ReLU in $a^{\langle 1 \rangle}$

$g_2 \rightarrow$ sigmoid in $\hat{y}^{\langle 1 \rangle}$

In general,

$$a^{\langle t \rangle} = g(w_{aa} a^{\langle t-1 \rangle} + w_{ax} x^{\langle t \rangle} + b_a)$$

hidden state

$$\hat{y}^{\langle t \rangle} = g(w_{ya} a^{\langle t \rangle} + b_y)$$

prediction

Simplified RNN notation:

$$a^{\langle t \rangle} = g(w_{aa} a^{\langle t-1 \rangle} + w_{ax} x^{\langle t \rangle} + b_a)$$

$$w_{aa} a^{\langle t-1 \rangle} + w_{ax} x^{\langle t \rangle} = w_a [a^{\langle t-1 \rangle}, x^{\langle t \rangle}]$$

Page No.
Date

$$100 \downarrow \begin{bmatrix} W_{aa} & \vdots & W_{am} \end{bmatrix} = W_a \downarrow (100, 10100)$$

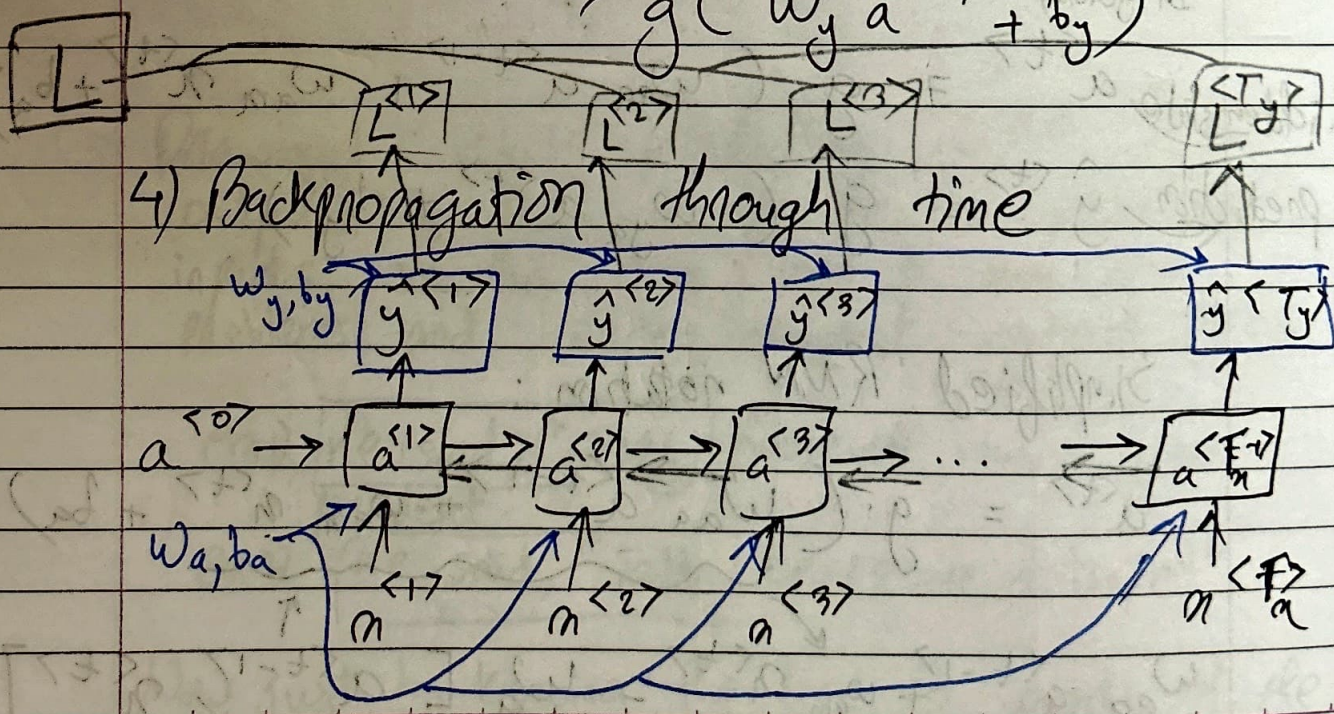
$\longleftrightarrow$ 100 $\quad$ $\longleftrightarrow$ 10000

$$\left[a^{\langle t-1 \rangle}, a^{\langle t \rangle}\right] = \begin{bmatrix} a^{\langle t-1 \rangle} \\ a^{\langle t \rangle} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 10000 \end{matrix} \quad \updownarrow 10100$$

$$\therefore \begin{bmatrix} W_{aa} & \vdots & W_{am} \end{bmatrix} \begin{bmatrix} a^{\langle t-1 \rangle} \\ a^{\langle t \rangle} \end{bmatrix} = W_{aa}\, a^{\langle t-1 \rangle} + W_{am}\, a^{\langle t \rangle}$$

also, $\hat{y}^{\langle t \rangle} = g(W_{ya}\, a^{\langle t \rangle} + b_y)$

$g(W_y\, a^{\langle t \rangle} + b_y)$



loss function:

$$L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1-y^{\langle t \rangle}) \log (1-\hat{y}^{\langle t \rangle})$$

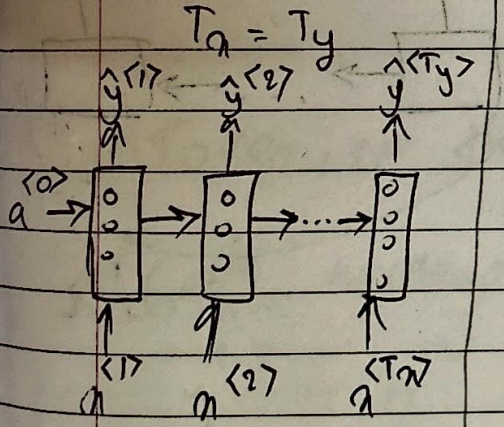
Overall loss,

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$

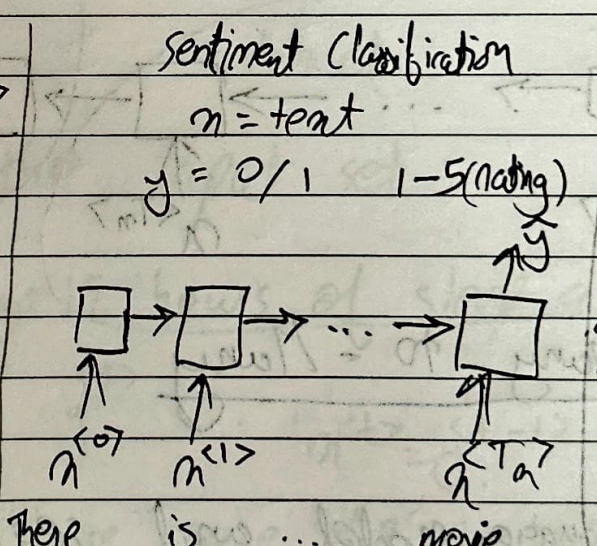
→ sum of individual losses.

* Backpropagation goes from right to left i.e. backwards in time, hence the name back propagation through time.

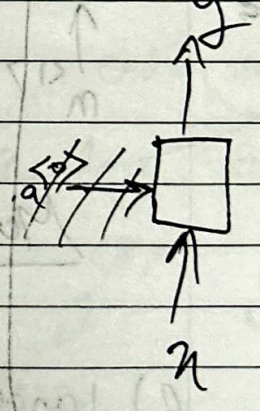
## 5) Different types of RNNs ($T_x \neq T_y$)




Many-to-Many




Many-to-one

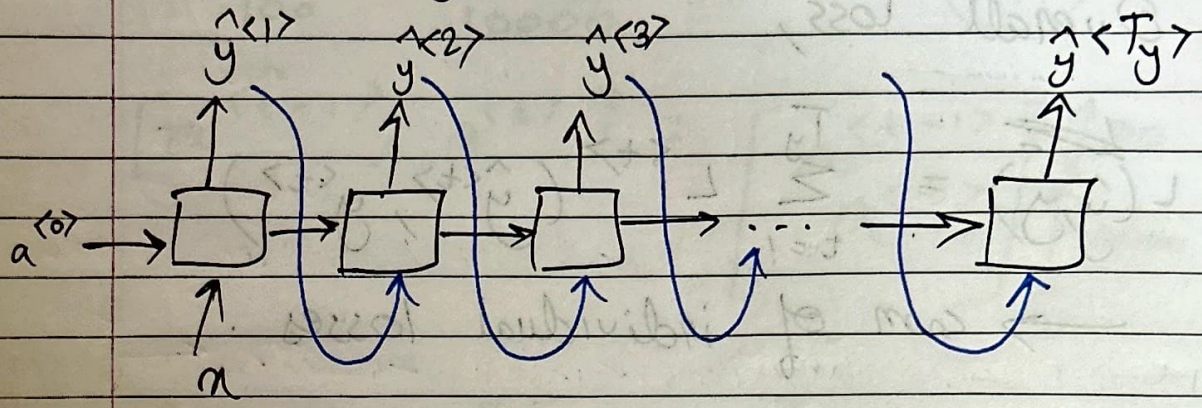



One-to-One

Page No.
Date

* Music generation

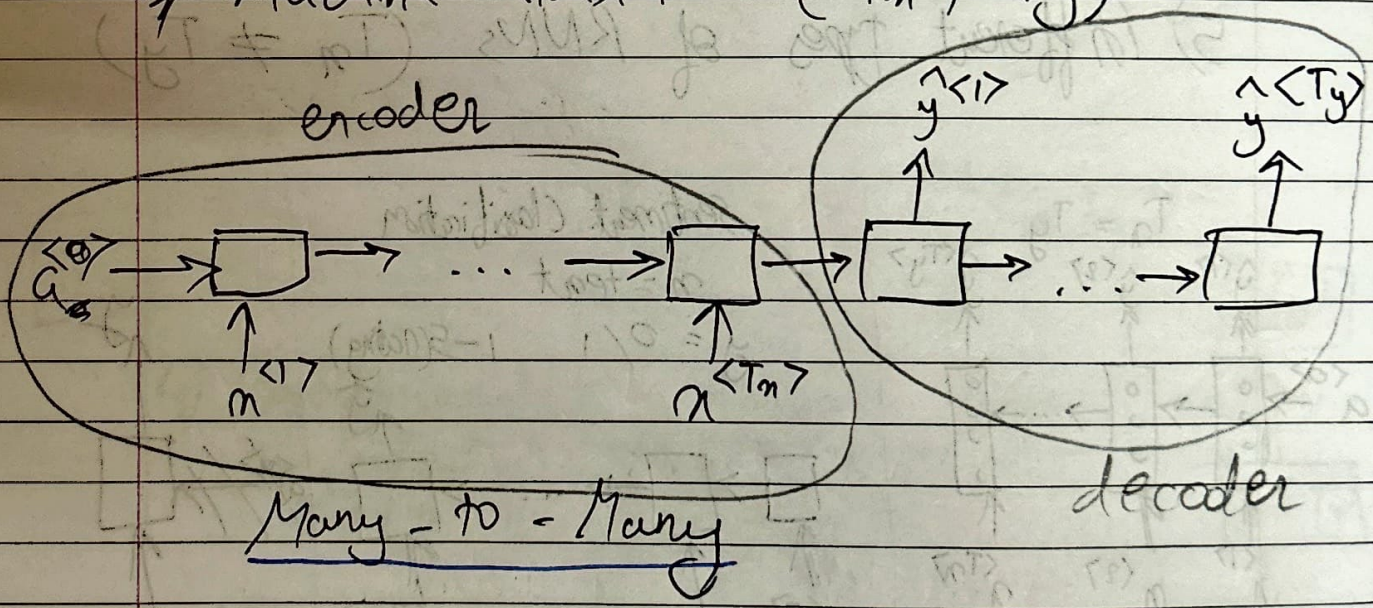
$n \rightarrow y^{\langle 1 \rangle} y^{\langle 2 \rangle} \cdots y^{\langle T_y \rangle}$




One - to - many

Also not $n$ can be null set here i.e. $n = \emptyset$

* Machine Translation ( $T_n \neq T_y$)




Many - to - Many

6) Language model and sequence generation :

↓

Speech Recognition :

The apple and pain salad.

The apple and pear salad.

$P(\text{sentence}) = ?$ $P(y^{<1>}, y^{<2>}, \dots y^{<T_y>}) = ?$

$P(\text{The apple and pain salad}) = 3.2 \times 10^{-13}$

$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$

↑ This one has more probability hence it will be produced as output.

Language modelling with an RNN

Training set: large corpus of english set.

dataset

Tokenize the input set

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

end of sentence

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $x^{<t>} = y^{<t-1>}$ $y^{<8>}$ $y^{<9>}$

In this case, we ignore the punctuation.

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

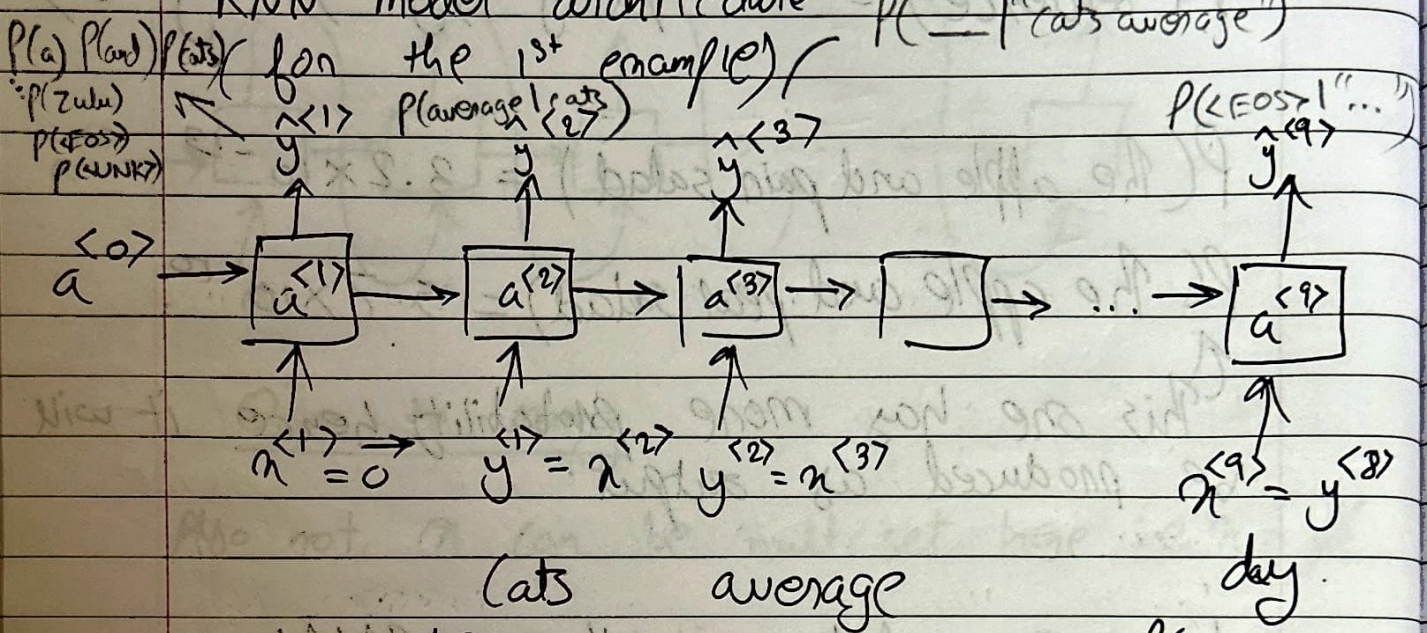
<UNK>

↳ unknown word since "Mau" won't be in our dictionary of most common 10000 words

Page No.
Date

Tokenization basically means taking the input sentence and map to the individual tokens/ individual words / individual vocab.

RNN model architecture:




Cats average day

i.e. given the first few words what is the P(next word)?

Cost function:

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

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

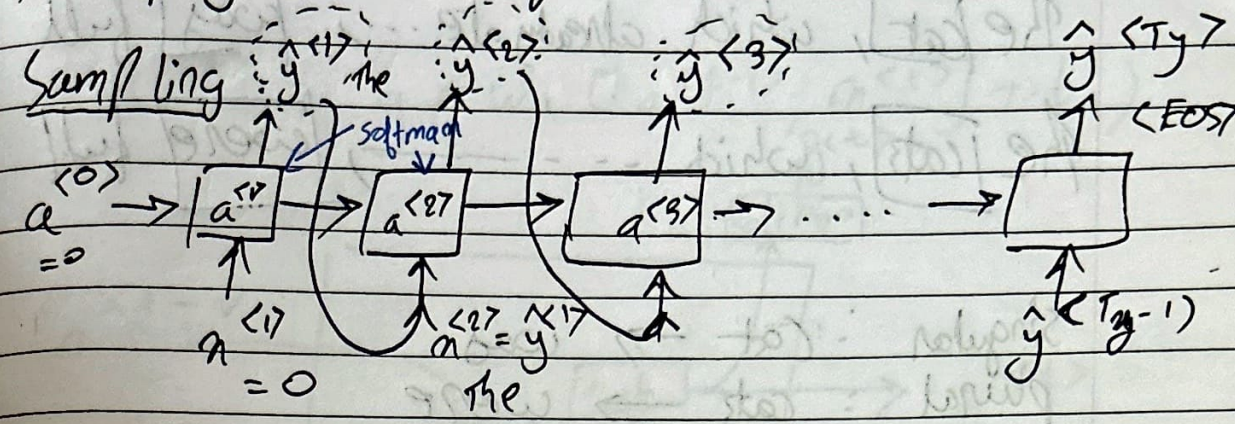
Note: Probability of three words in sentence -

$$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) * P(y^{<2>} | y^{<1>}) * P(y^{<3>} | y^{<1>}, y^{<2>})$$

Page No.
Date

## 7) Sampling Novel sequences

Sampling




$y^{<1>} \rightarrow P(a), P(aaron) \cdots P(zulu)\ P(<\text{UNK}>)$

$y^{<2>} \rightarrow P(\_ \mid the)$

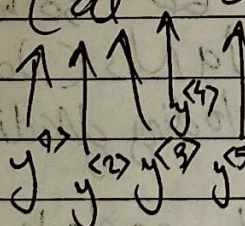
Sampling is generating words based on the probabilities and can be implemented as:

→ np.random.choice

## Character-level language model

Vocabulary = [a, b, c, ..., z, ␣, ?, ;, 0...9, A, ..., Z]

eg: Cat average ...




character level models one character at a time ∴ less effective.

Pros: No need to assign <UNK> to any word as all the words can be built using vocabulary.

Cons: Computationally expensive to train and also

Page No.
Date

8) Vanishing gradients with RNNs

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

The cats, which ---------, were full

singular : cat → was
plural : cats → were

The RNN needs to remember whether the subject was singular or plural but it turns out that even in forward propagation the P(was | ___) or P(were | ___) will depend on some of the previous words more as compared to all the way beginning of the sentence words. Similarly while calculation derivatives in back propagation, the derivatives of the left-most nodes/parameters will less likely depend on those of the ~~the~~ right most ones as it slowly vanishes and becomes less important to compute.

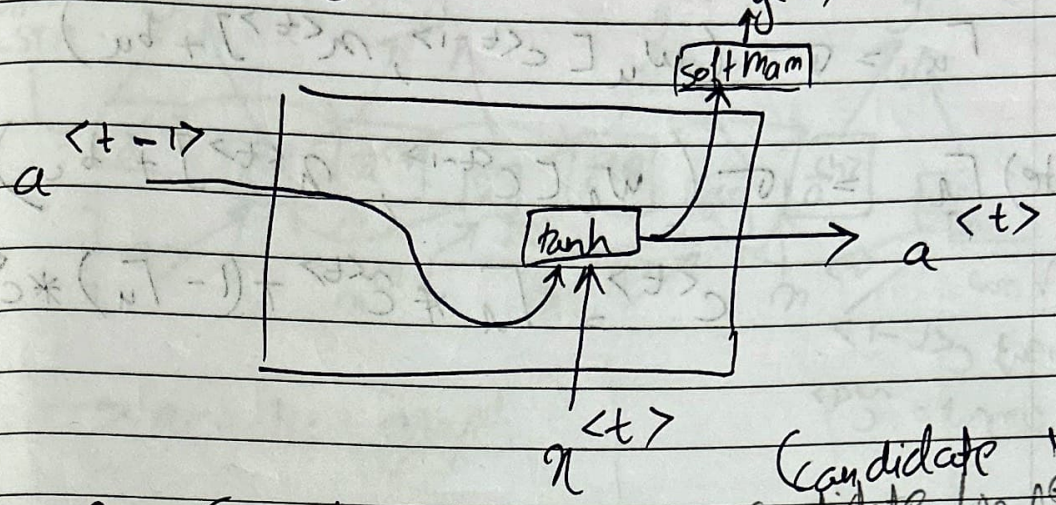
We faced a problem "Exploding gradients" in Deep NNs which can be solved using gradient clipping. NaNs (Not a number) is resulted in case of Exploding gradients

Also, it is easier to spot exploding gradients in Deep NNs than vanishing gradients as exploding gradients give us very high numerical values which affect the parameter by a high factor so easier to spot.

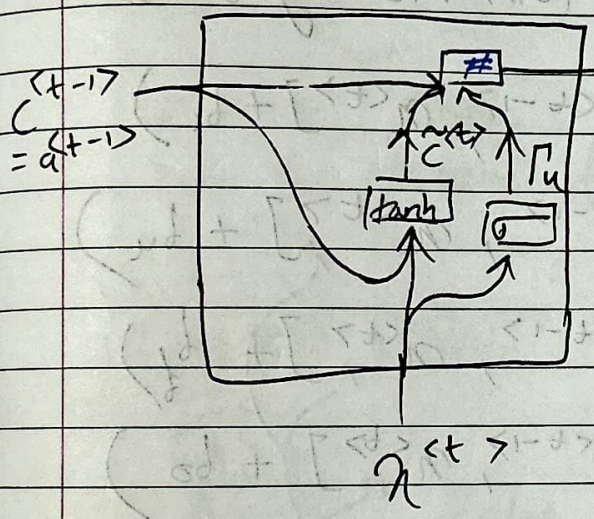
Page No.
Date / /

# 9) GRU : gated Recurrent unit

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




## GRU (simplified)




candidate value
candidate for replacing memory cell

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

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

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

$\Gamma_u$ : capital gamma, update gate

"0" or "1" for most of the values

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

element-wise multiplication

$\Gamma_u = 1$, $c^{<t>} = 1$ — $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ ...

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

# LSTM > GRU

$c^{\langle t \rangle}$ → cell state, $\tilde{c}^{\langle t \rangle}$ → candidate value ~~state~~

## Full GRU

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

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

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

tells how relevant $c^{\langle t-1 \rangle}$ is to compute $\tilde{c}^{\langle t \rangle}$

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

## 10) LSTM (long short term memory units)

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

dim → no. of activations

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

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

output gate: $\Gamma_o = \sigma\left(W_o [a^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o\right)$

~~$c^{\langle t \rangle} = \Gamma_u * \tanh c^{\langle t \rangle}$~~

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

$$a^{\langle t \rangle} = \Gamma_o * \tanh c^{\langle t \rangle}$$

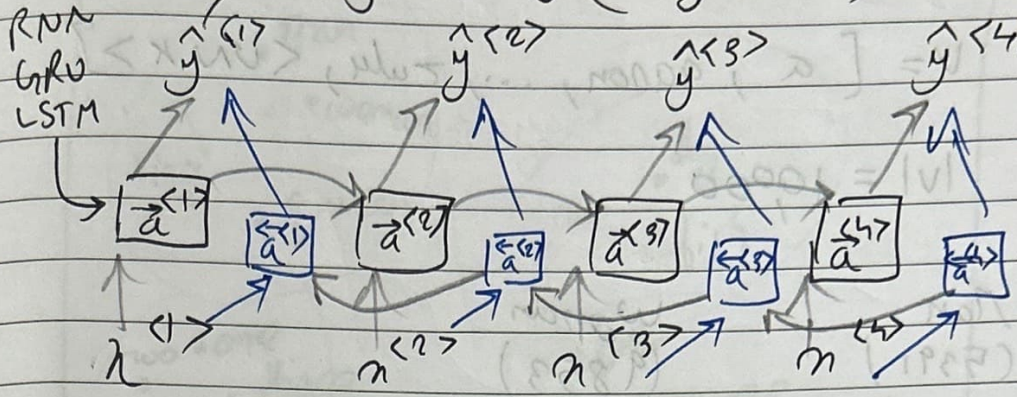
N/A: peephole connection: $c^{\langle t-1 \rangle}$

from assignment: $y^{\langle t \rangle}_{pred} = \text{softmax}\left(W_y a^{\langle t \rangle} + b_y\right)$

Page No.
Date

## 11) Bidirectional RNN

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



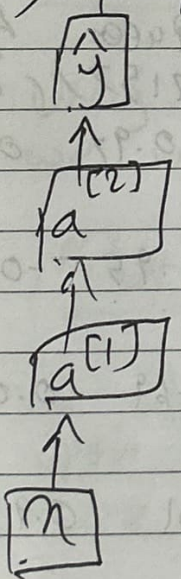

→ acyclic graph

He said "Teddy Roosevelt ..."

Hence, teddy will be predicted to be a person from the entire sequence.

## 12) Deep RNNs

$a^{[l]<t>}$ — layer $l$, time $t$




Refer ss from chat