Camlin | Page
Date / /

Sequence Modeling

COURSE - 05

Camlin Page
Date / /

SEQUENCE DATA :-

1. Speech Recognition
2. Music generation
3. Sentiment Classification
4. DNA Sequence Analysis

Motivating Example :=

Problem: Name entity recognition. Recognize names in below sentences.

X: Harry Potter & Hermone Granger invented a new spell.
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ ... $x^{<t>}$ ... $x^{<9>}$

$T_x = 9 = \text{length}$

Y: 1 1 0 1 0 0 0 0 0
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<9>}$

$T_y = 9 = \text{length}$

$X^{(i)<t>}$ = i^{th} training sample, t^{th} word

$T_x^{(i)}$ = length of i^{th} training sample.

$y^{(i)<t>}$ = t^{th} element in i^{th} training sample in OUTPUT SEQUENCE.

$T_y^{(i)}$ = length of i^{th} OUTPUT SAMPLE.

Representing Word :=

x: Harry Potter & Hermione Granger invented a ___
$x^{<1>}$ $x^{<2>}$ $x^{<5>}$ $x^{<6>}$ ___ .

Vocabulary :

a	1
aaron	2
and	→ 367
harry	→ 4075
potter	→ 6830
zulu	→ 10000

In this example we are using vocabulary of size = 10K.

$x^{<1>}$ → [0, 0, ..., 1, 0, 0, 0] → 4075 (one-Hot)

$x^{<2>}$ → [0, 0, 0, ..., 6830, 0, ...]

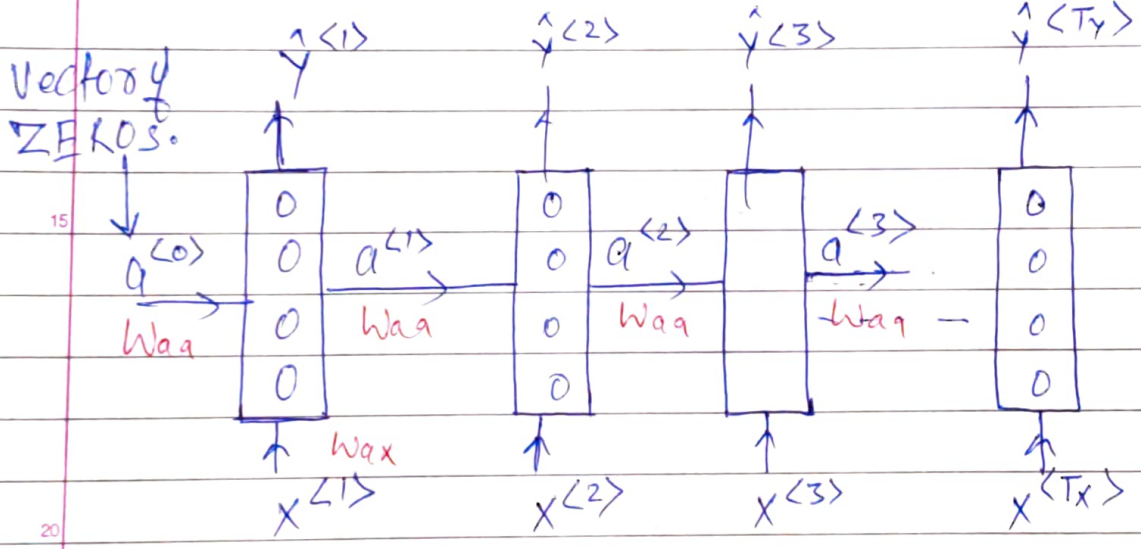
$x^{<3>}$ → [0, 1, 0, 0, ..., 0] → 367

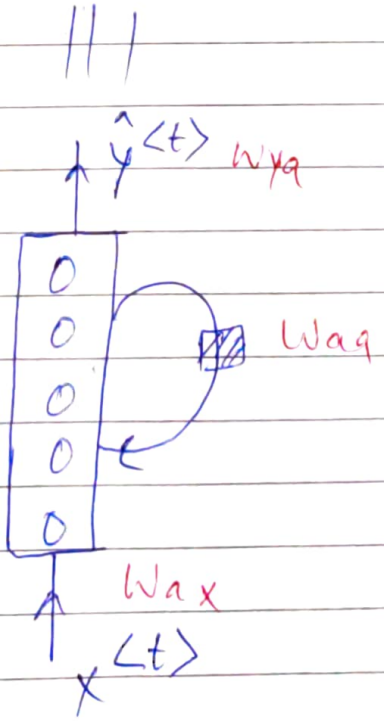
Why Not Standard Neural Network :-

Problems :- 1. Input / Output can be different lengths in a different examples.

2. Doesn't share features learned across different positions of text.

Recurrent Neural Network :-





(X) Waa, Wax & Way all are same across the network.

Let's Apply Activation functions :-

$$a^{<1>} = g(W_{aa} * a^{<0>} + W_{ax} x^{<1>} + b_a)$$

$$y^{<1>} = g_2(W_{ya} * a^{<1>} + b_y)$$

g_1 = tanh/Relu g_2 = sigmoid

Generalizing :-

$$a^{<t>} = g_1(W_{aa} * a^{<t>} + W_{ax} * x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ya} * a^{<t>} + b_y)$$

Simplifying :-

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

$W_a = 100 \downarrow \quad [W_{aa} \;\vdots\; W_{ax}] = W_a$ (100, 10100)

100 → 1000 →

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 1000 \end{matrix} \quad \updownarrow 10100$$

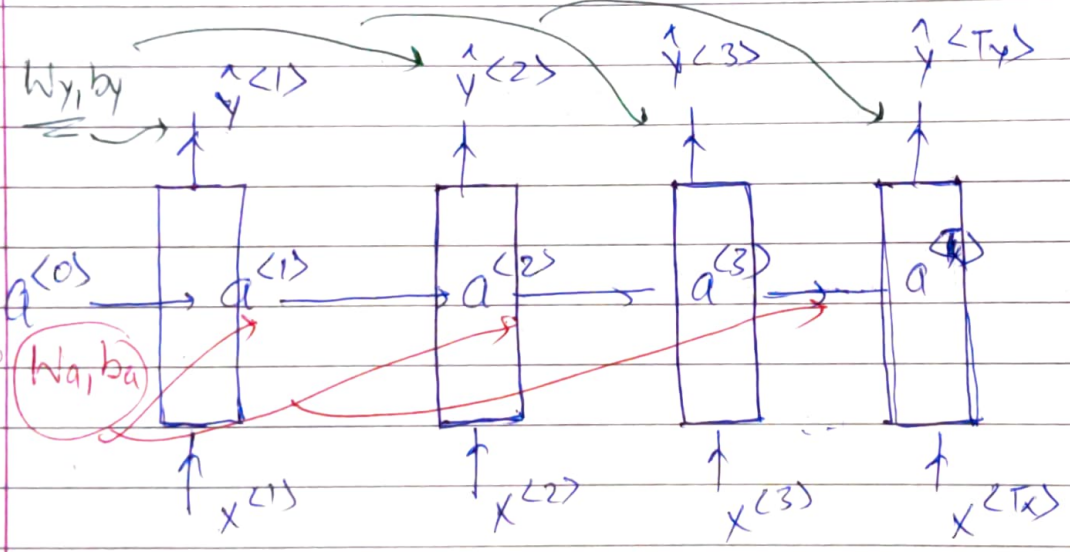
We can check

$$[W_{aa} \mid W_{ax}] \begin{bmatrix} a^{\langle t-1 \rangle} \\ x^{\langle t \rangle} \end{bmatrix} = W_{aa} * a^{\langle t-1 \rangle} + W_{ax} * x^{\langle t \rangle}$$

In similar way :-

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

Back Propogation RNN :-



(W_a, b_a) (W_y, b_y) weights are shared

Elementwise loss function

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

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

Different Types · RNN

T_x = Input length | T_y = Output length.

$T_x = T_y$ = many to many

$T_y = 1$, $T_x > 1$ = many to one (Sentiment Classi)

$T_x = 1$, $T_y > 1$ = One to many

(music generation)

[Language Model]

[Speech Recognition]

S1: The apple & pair salad.

S2: The apple & pear salad.

$P(S1) = 3.2 \times 10^{-13}$

$P(S2) = 5.7 \times 10^{-10}$

S2 is much more likely.

Sentence → [Language Model] → P(sentence)

Input → Output

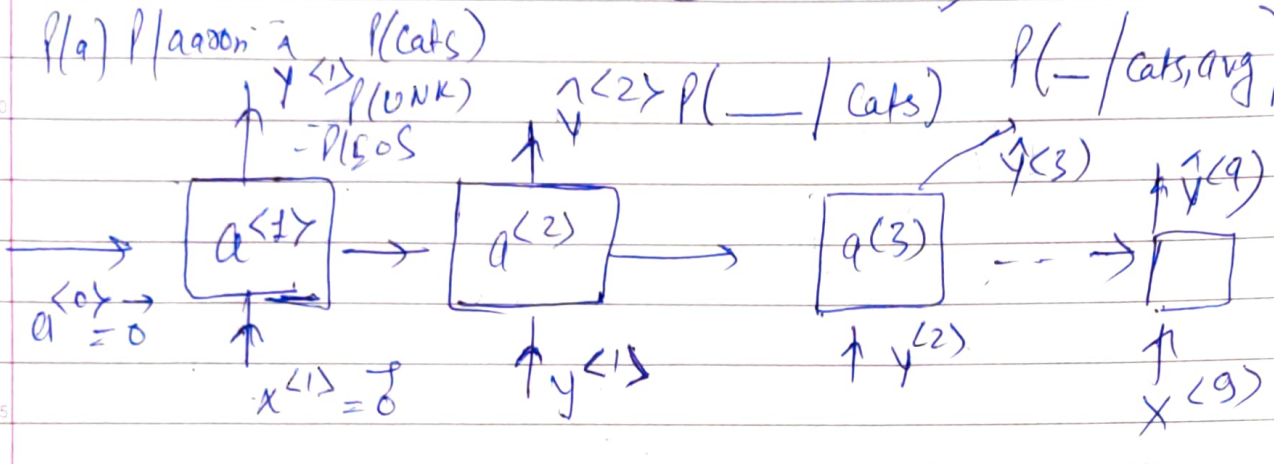
END of Sentence.

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

$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ ... $y^{<6>}$ $y^{<9>}$

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

<UNK> (unknown)



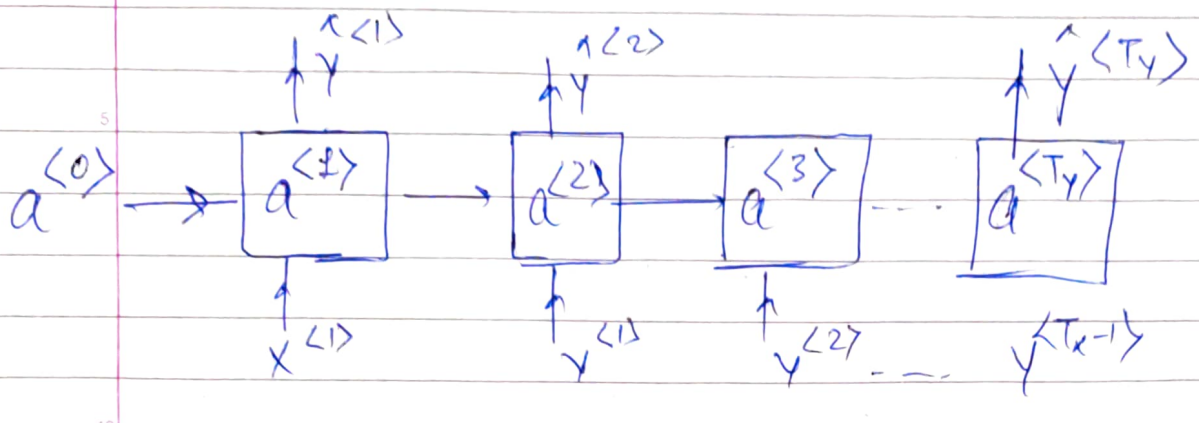
$= y^{<8>} = day$

Cost function

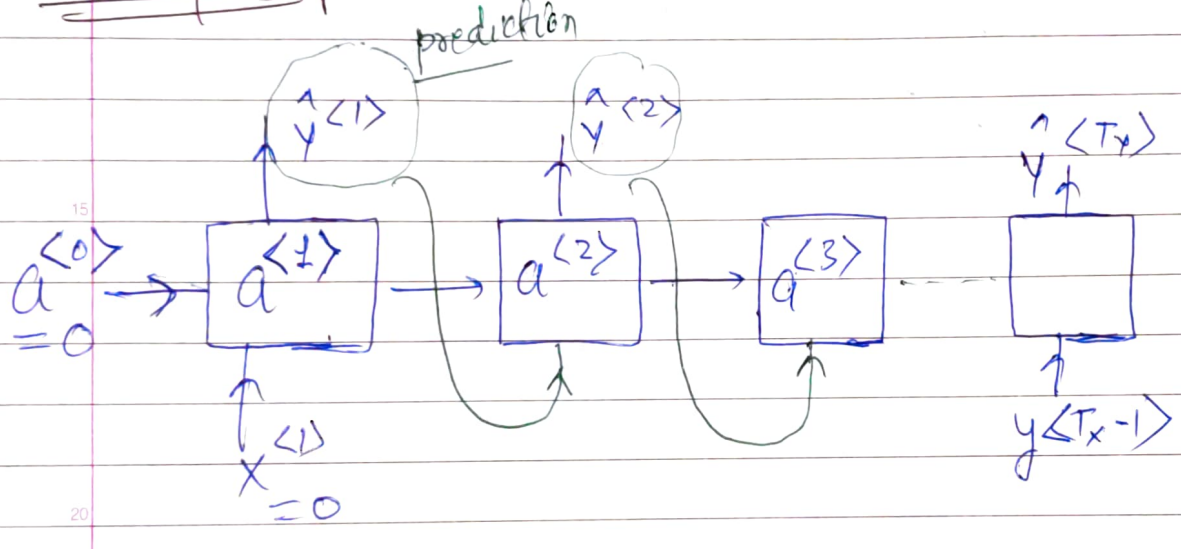
$$\mathcal{L}(\hat{y}^{t}, y^{<t>}) = -\sum_{i} y_i^{<t>} \log \hat{y}_i^{<t>}$$

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

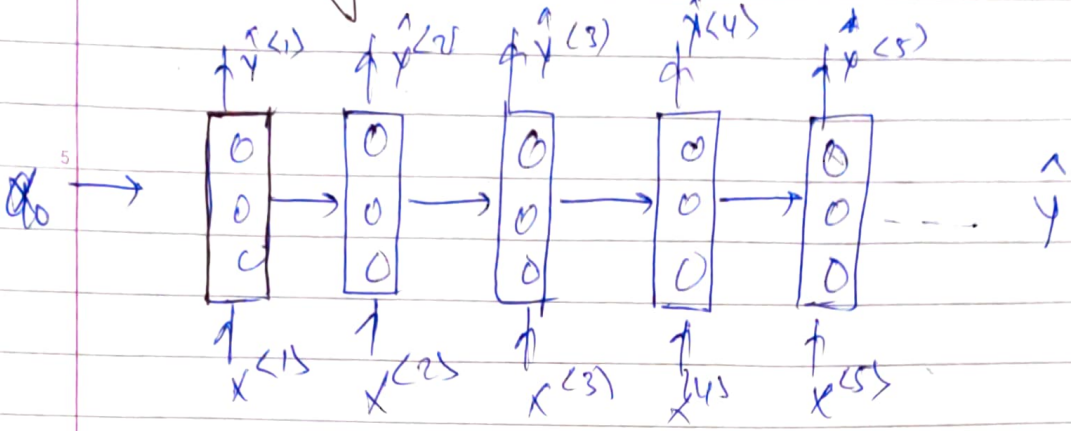
Sampling a Sequence from trained RNN



Sampling :-



Vanishing Gradients with RNN



Problem:- When the Gradients are being propogated back in time all the way to the initial layer. The gradients coming from the deeper layer have to go through CONTINOUS MATRIX MAULTIPLICATIONS because of the Chain Rule. If they have small values (<1) they shrink exponentia-lly until they vanish & make it impossible that model can learn.

↳ Gradient got Vanish. (Vanishing Gradient)

In the same way if (values >1) multiplying over the time it become NaN.

Gradient Explode (Exploding Gradient).

Camlin Page
Date / /

Solution :- Gradient Clipping

If $\|g\| > \text{threshold}$

$g \leftarrow \frac{\text{threshold} \times g}{\|g\|}$

} Exploading Gradient

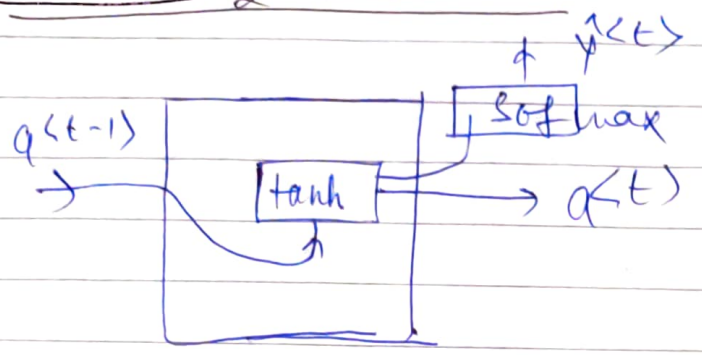
Vanishing Gradient Solution :-

Redesign RNN :

The standard RNN operates in such a way that hidden state activation are influenced by the other local activation closest to them correspond as "Short Term Memory", while network weights are influenced by the computations that take over entire long sequences correspond to "long-term memory". Hence RNN redesign known as "LSTM", "GRU" etc.

—: GRU :- [Simplified Version]

RNN Unit Look like :-



GRU :-

C = memory cell

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

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

Gate name GAMA U $\Rightarrow$ Γ_u = (0 or 1)

in practice Γ_u = sigmoid function which is most of the time 0 or 1

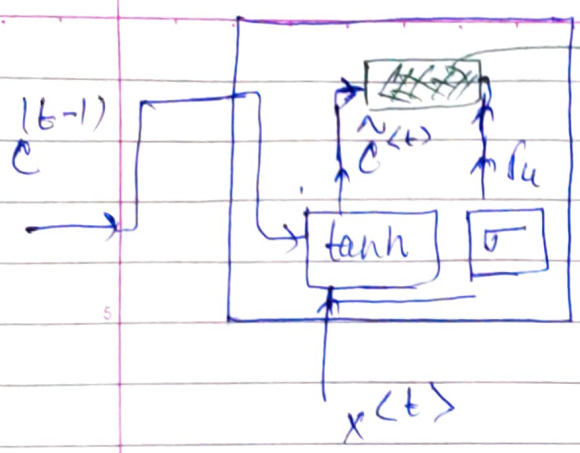
$$\Gamma_u = \sigma\left(W_u \left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

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

↳ element wise multip

it says $\Gamma_u = 1$ update gate by $\tilde{c}^{\langle t \rangle}$

& $\Gamma_u = 0$ take old value (Not Update).



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

Full GRU := Full GRU introduce one more SIGMOID GATE.

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

$\equiv$

Γ_r = Intitution of Γ_r is tell how much relevance of $c^{\langle t-1 \rangle}$ earlier it was full.

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

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

↳ Element wise multiply.

LSTM

Putting LSTM same as GRU in equation.

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

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

Unlike GRU where we $(\Gamma_u \& 1-\Gamma_u)$ we have two gates more

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

↳ forget gates.

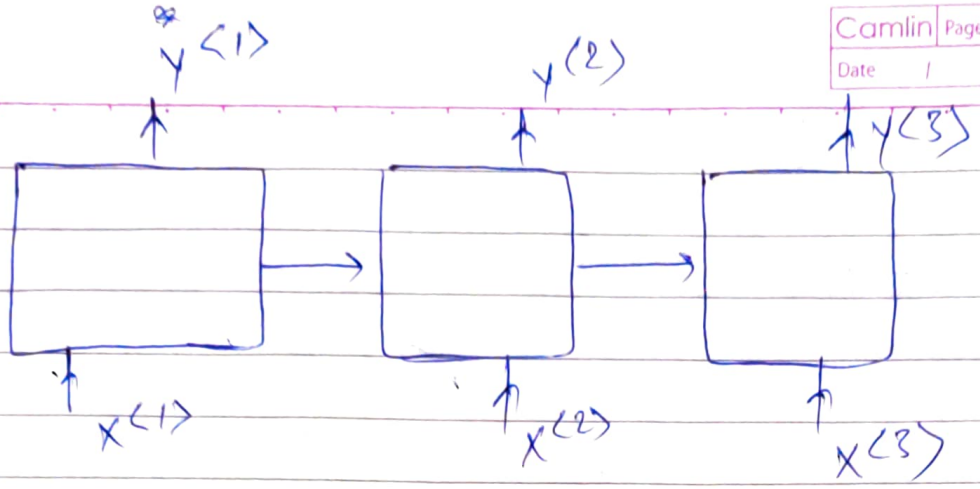
$$\Gamma_o = \sigma\left(W_o \left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right)$$

↳ output gate

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

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

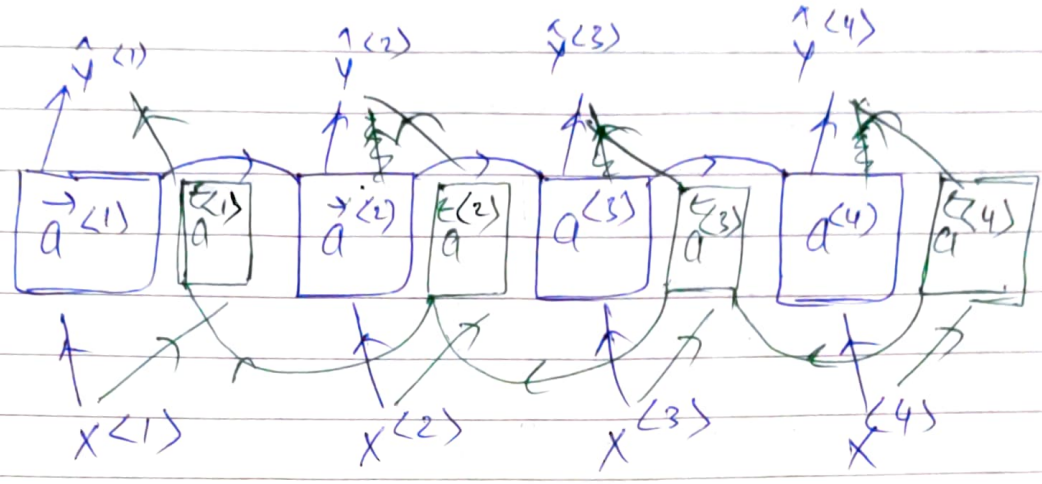
All these equation is one LSTM Unit.



(*) Advantage of GRU simple model so much bigger network can be used. — ANDREW NG.

But LSTM is more full. LSTM is more generally default choice.

Bi Directional-RNN :=



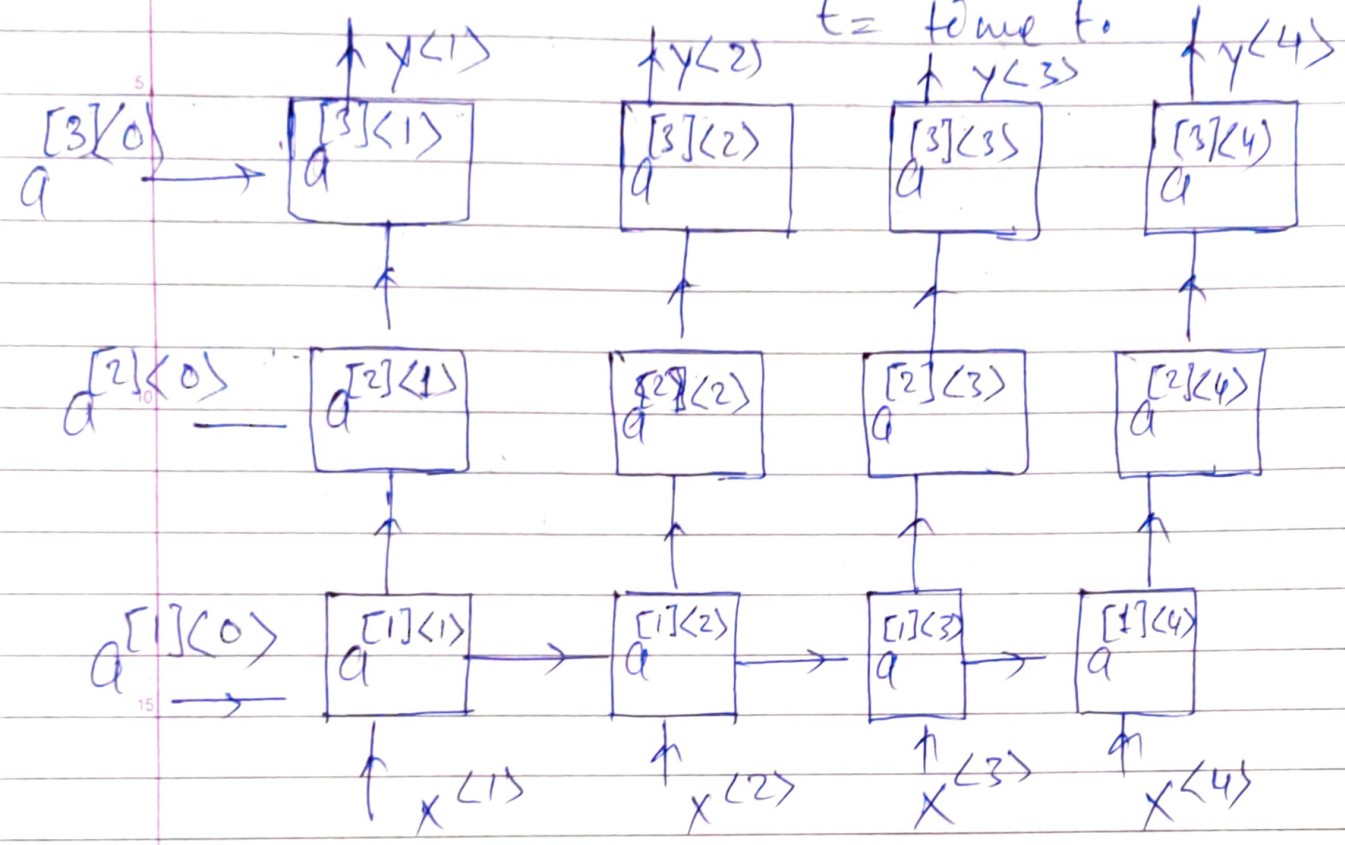
$$\hat{y}^{\langle t\rangle} = g\left(W_y \left[\vec{a}^{\langle t\rangle}, \overleftarrow{a}^{\langle t\rangle}\right] + b_y\right)$$

Deep RNN !=

$a^{[l]\langle t\rangle}$

l = lth layer

t = time t.



WEEK-02

[INTRO WORD EMBEDDINGS]

V = [a, aaron ... zulu, <UNK>] ||V|| = 10K

1- hot representation (Man - 5391) (Woman = 9853)

↳ Man = [0 0 0 0 1 0] → 5391

[0 0 0 0 ⋮ 1 0] → 9853

- feature representation: Word Embedding

	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	-.95	.97	.00	.01
Royal	.01	.02	.93	.95	-.01	.00
Age	.03	.02	0.7	.69	.03	102
Food	.04	0.01	.02	.01	.95	.97
Size	⋮	⋮	⋮		⋮	
Cost	300	300	300			
	e_{5391}	e_{9853}				

300 (.9)

Orange & Apple feature look | pretty similar.

Transfer Learning Word Embeddings :-

1. Learn word embeddings from large text corpus - (1-100 Billion words)
2. Transfer embedding to new task with smaller training set. (100k words)
3. Optional :- Continue to fine out the word embeddings with new data

:- Analogies :-

	(5391) Man	(9853) Woman	(4914) King	(7157) Queen	(456) Apple	(6257) Orange
Gender	-1	1	-.95	.97	.00	.01
Royal	.01	.02	.93	.95	.01	.00
Age	.03	.02	.70	.69	.03	-.02
food	.09	.01	.02	.01	.95	.97
	e_{5391}	e_{9853}	e_{4914}	e_{7157}	e_{456}	e_{6852}

$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$

$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ \vdots \end{bmatrix}$

(Mikolov et. al. 2013)

$e_{man} - e_{woman} \approx e_{king} - e_?$

Find word w $\arg\max$ $sim(e_w, e_{king} - e_{man} + e_{woman})$

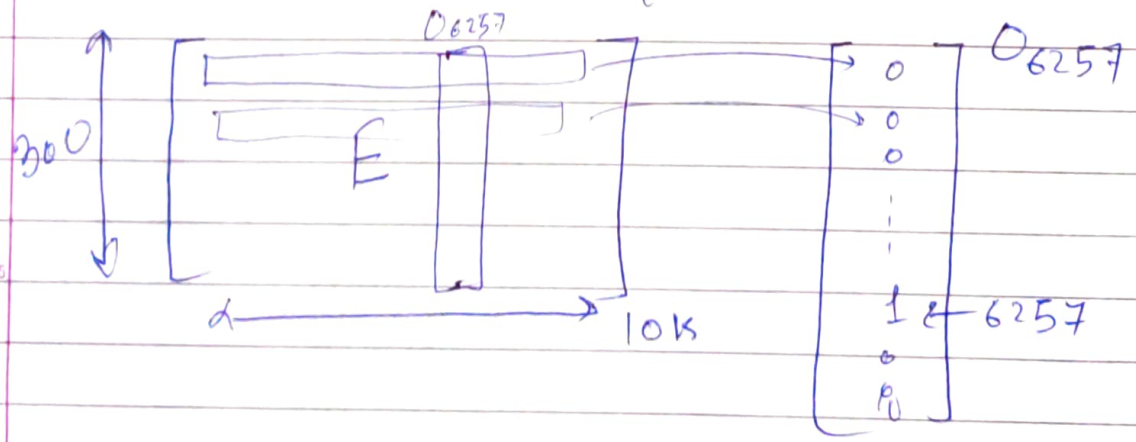
t-SNE ($\mathbb{R}^N \rightarrow \mathbb{R}^2$)

$sim(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$ [Cosine Similarity]

[Learning Embedding Matrix]

learn Word Embedding matrix

a aaron .. orange (6257) .. zulu <UNK>



$E \cdot O_{6257} =$  $= e_j$ = embedding for word j.

We initialize E randomly & use Gradient Descent to learn the parameter of E (300 x 10000) dimension matrix.

Specific ALGO Learn matrix E.

[Neural Language Model]

I want a glass of orange ——.

(4343) (9665) (1) (3852) (6163) (6257)

(one-hot) (Embeding Mat)

I $O_{4343} \longrightarrow E \longrightarrow e_{4343}$

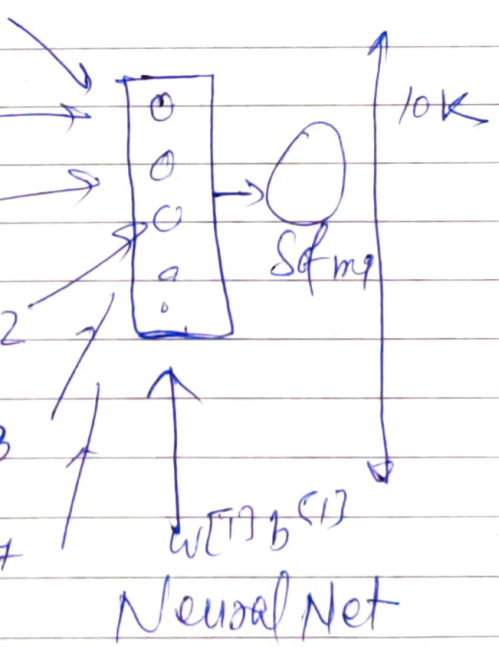
want $O_{9665} \longrightarrow E \longrightarrow e_{9665}$

a $O_1 \longrightarrow E \longrightarrow e_1$

glass $O_{3852} \longrightarrow E \longrightarrow e_{3852}$

of $O_{6163} \longrightarrow E \longrightarrow e_{6163}$

orange. $O_{6257} \longrightarrow E \longrightarrow e_{6257}$



Hyperparameter last k- words. k=4

in this case input to neural net

4 x 300 (Stack word) = 1200

know use backprop to maximize the prediction. This way we will learn good E (Embedding matrix).

Hyper param k = Context

→ Last 4 words → Last 1 word
→ 4 ∞ words on left & right.

Word 2 Vec

Simpler & less computational way to learn (E) embedding word matrix.

X: I want a glass of orange juice to go along with my cereal.

Choose random word say = orange then in ±5 vicinity form below pair

context	Target
orange	juice
orange	glass
orange	my
orange	go

[Skip Gram Model]

Vocab size = 10k

Context c("orange) ⟶ Target("juice)

Learn this mapping.

$O_c \xrightarrow{E} e_c = (E * O_c) \rightarrow$ Softmax Unit $\rightarrow \hat{y}$

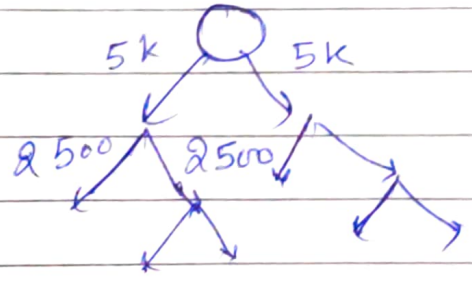
$$\text{Softmax} : p(t/c) = \frac{e^{\theta_t^T * e_c}}{\sum_{j=1}^{10k} e^{\theta_t^T * e_c}}$$

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{10k} y_i \log \hat{y}_i = \text{Loss function.}$$

$y = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$ ← 48 / 10k one hot $\qquad \hat{y} = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

⊗ Problem: Softmax is very slow to compute

⟹ Use Hierarchical Softmax classifier.



Negative Sampling :=

X: I want a glass of orange juice to go along with my cereal.

[look around ± 5]

Context	Word	Target
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

↑ pick random word from dictionary for -ve word.

k times -ve example.
k = 5 ~ 20 Smaller Dataset

k = 2, 5 larger Data set.

X | Y

$P(y=1 / c, t) = \sigma(\theta_t^T e_c)$

every +ve example we have k -ve example.

Camlin Page
Date / /

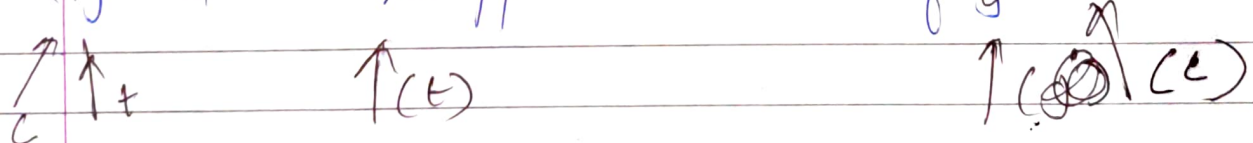
we train only on (k+1) samples a Neural Network. [k negative, 1 positive]

[Glove Word Vectors]

X: I want a glass of orange juice to go along with my cereal.

Till now we were choosing C,t in close vicinity ±5. But for GLOVE,

Xij = times j appears in context of i.



Xij = is count how often words appear close to each other.