RNN

PAGE No.
DATE / /

*Application

* • RNN works very well with Sequence of Data as your Input.
eg:- NLP (SPAM, Time series Data, Stock forecasting) etc.

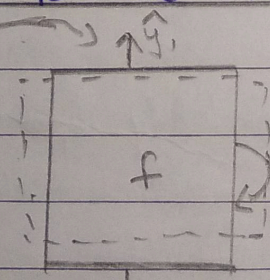
eg:
(1) NLP
↳ BoW, TF-IDF, word2vec
convert the Sentence into vector

NLP
↳
eg:- Google Assistant, Amazon Alexa

eg:- Time Series
RNN will help to predict [Here RNN work very well]

eg: Google Image Search
eg: Google Translator
eg: Image Captioning

* RNN Forward Propogation with Time



← Hidden layer here we have any no. of hidden neurons
~ we also get o/p with respect to time

• NLP use case Sentiment Analysis

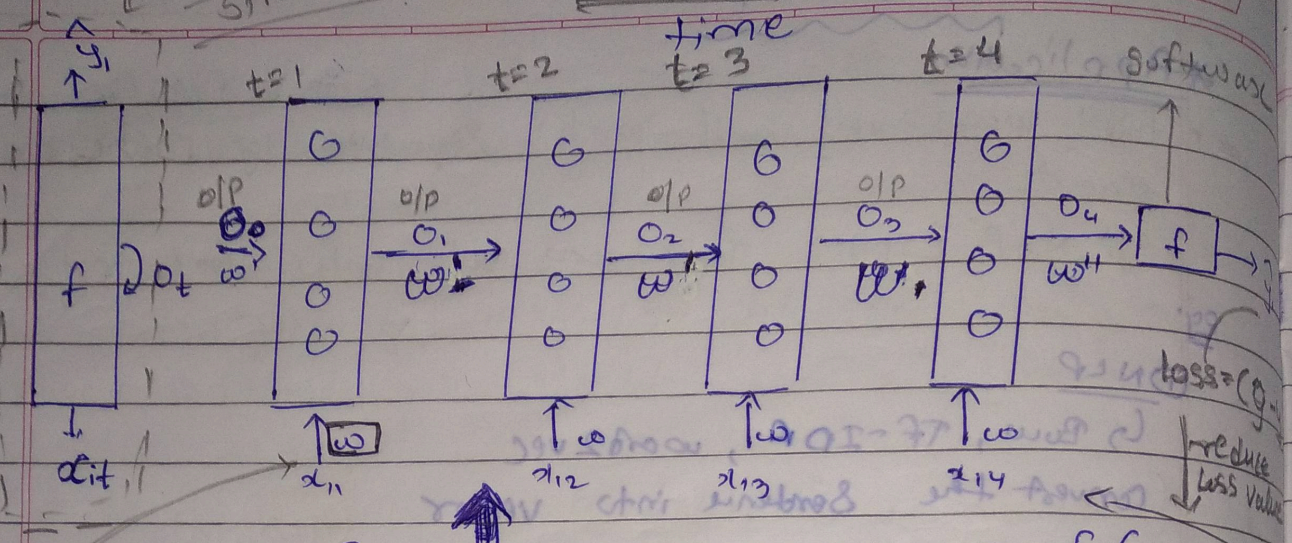
i/p → $x_i = \langle x_{i1}, x_{i2}, x_{i3}, x_{i4} \rangle$ (1, 2, 3, 4)

o/p → y_i

Our RNN at each and every time at t=1, will preprocess 1 word, then at t=2 .... preprocess second word ...

RNN Structure

Forward Propagation over time



* forward Propagation *

output 1 $\Rightarrow O_1 = f(w_i, x_w)$

• NLP use case Sentiment Analysis.

I/p x weight → then we give to Hidden Layer

$x_1 = \langle x_{11}, x_{12}, x_{13}, x_{14} \rangle$ ← Suppose I have these 4 words

y_1 ← o/p

At time t=1 my first word goes inside this particular hidden layer (x_{11})

Note: once representing the word in Vector

getting o/p → forward prop

$\rightarrow O_1 = f(x_{11} w + 0 w')$ ← function of O_1

$\rightarrow$ ~~$O_1 = f(x_i * w)$~~ ~~f ← function (tanh, sigmoid, etc.)~~

$\rightarrow O_2 = f(x_{12} w + O_1 w_1)$ # O_2 is completely dependent on x_{12} & O_1

$\rightarrow O_3 = f(x_{13} w + O_2 w')$

$\rightarrow O_4 = f(x_{14} w + O_3 w_1)$

Note :- # Sequence information is always Maintained, it is not discarded.

DATE

* Back Propagation (weights gets updated continuously w.r.t time)

→ $\frac{\partial L}{\partial \hat{y}} , \frac{\partial L}{\partial w''} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}_1}{\partial w''}$ ... (chain rule)

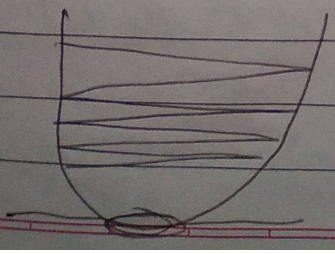
$w'' = w'' - \frac{\partial L}{\partial w''}$

→ $\frac{\partial L}{\partial \hat{y}_1} , \boxed{\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{O_4} \cdot \frac{\partial O_4}{\partial w}}$ ... (chain rule)

* Problem in Simple Recurrent Neural Network

Vanishing Gradient ⇒ The derivate of Sigmoid is between 0 to 1, now suppose one here we have calculated derivate function, it will be ~~right~~ between 0 to 1, but as we go towards this w, this value becomes very smaller value itself as the derivative is continuously happening, derivative of all ~~these~~ these particular weights this will become a very very small value (w), when it becomes a very very small value whenever we do an updatation of this particular weights, it will be negligible, then we will never converge to Global minima.

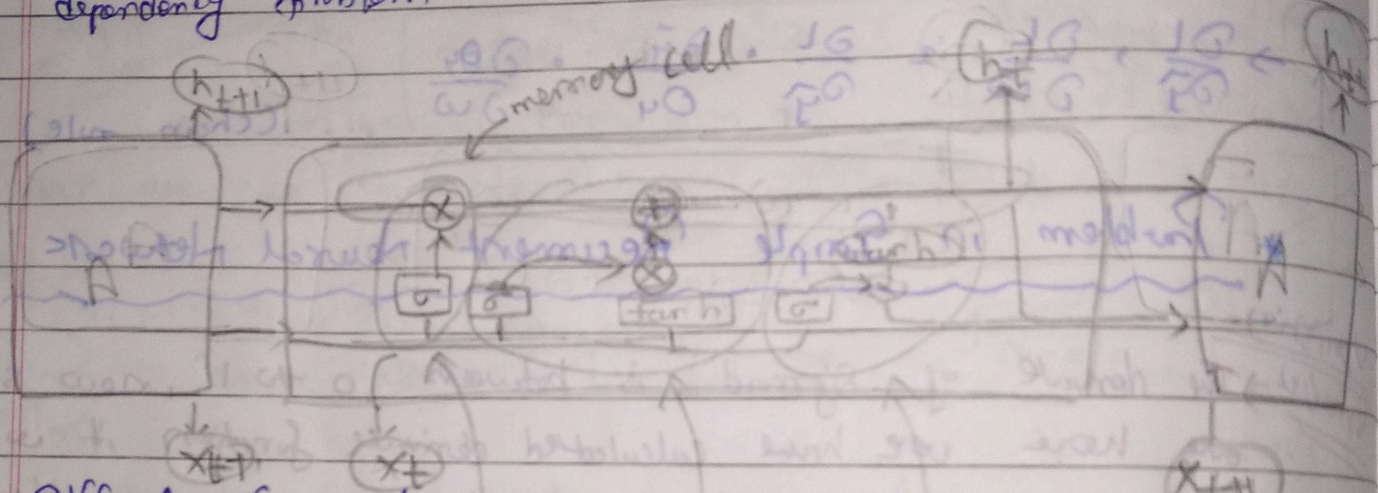
Exploding ⇒ If we are using other than sigmoid like ReLU, 1 > 0 (suppose my derivate is greater than 1), so we go ahead towards right to left, it creates Exploding Gradient problem, when this [w] changes will happen so big that it will never reach the global minima point



→ To solve this problem we use LSTM RNN (Long short term memory recurrent neural network)

* LSTM

- LSTM are explicitly designed to avoid the long-term dependency problem.



Different Components

1. Memory cell
2. Forget gate
3. I/P gate
4. o/p gate

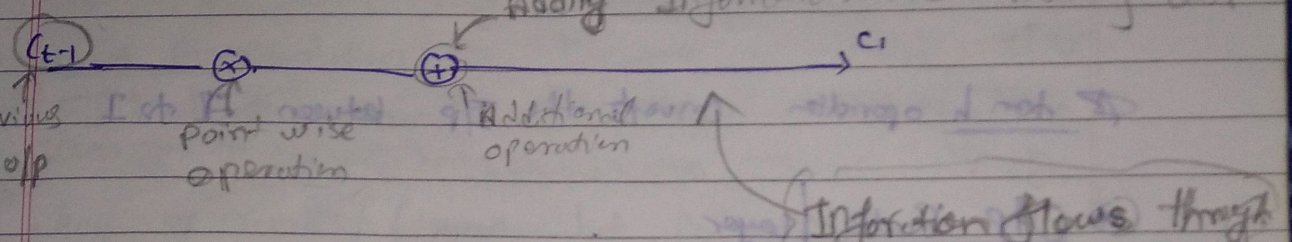
* Notation

- ▭ = Neural Network layer
- ○ = Pointwise operation
- → = vector Transfer
- ⟶ = Concatenate
- ⟶ = Copy

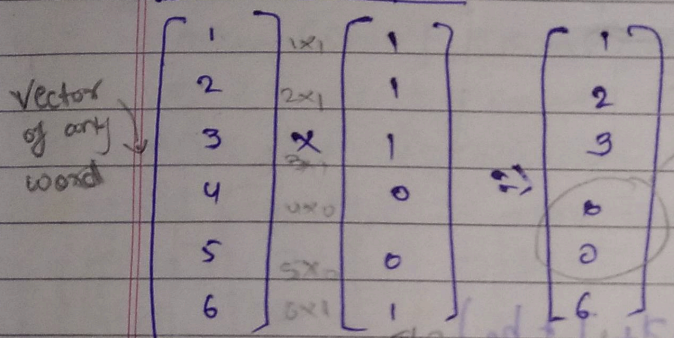
① Memory cell

→ Memory cell is used to ~~restrict~~ remember and forget things, ~~how do~~

→ How do we remember and forget is based on the context of the input

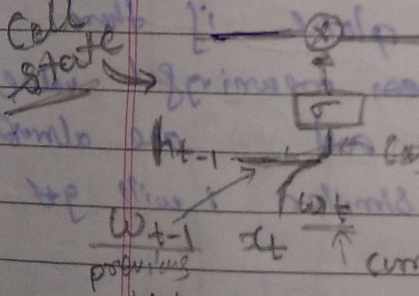


Point wise operation



0, 0 is basically forgetting this particular information. One of the information the model will forget because the context has changed.

② forget ~~retl~~ Gate



Sigmoid activation function

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

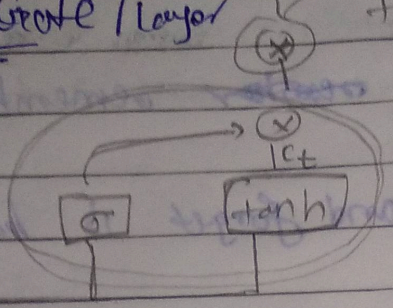
bias

y = mx + c

* σ ~~is~~ transform the i/p between 0 to 1

→ forget Gate ~~means~~ It either lets something in or doesn't let it in, which means it helps us the forget gate, it helps in defining what information to forget what information to remember

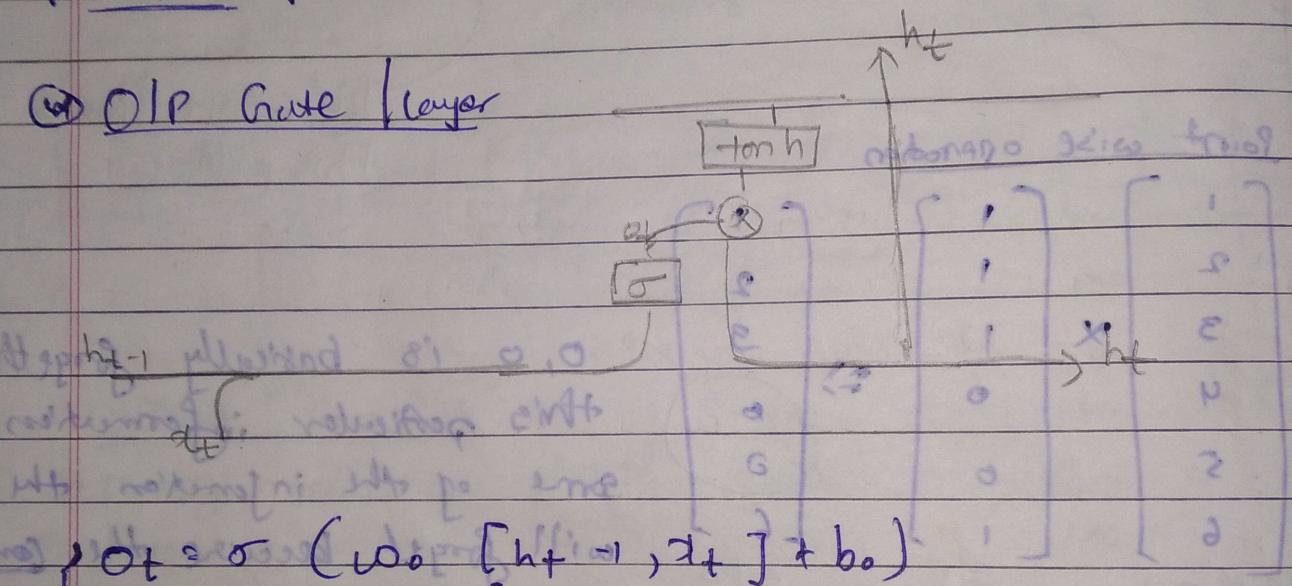
3) I/P Gate / Layer — Adding Information to the memory cell



$$\begin{cases} i_t = \sigma(w_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t = \tanh(w_c \cdot [h_{t-1}, x_t] + b_c) \end{cases}$$

* tanh operation converts the I/P between -1 to 1

4) O/P Gate / Layer



$$\begin{cases} o_t = \sigma(w_o [h_{t-1}, x_t] + b_o) \\ h_t = o_t * \tanh(C_t) \end{cases}$$

LSTM

Note:- If there is New Meaningful information then only this concatenation this operation will take place if there is no meaningful operation, no meaningful value that basically means my this I/P x_{t-1} and x_t are almost similar, then all my vector Similar, i will get all value as [1, 1, 1, 1.]

Word Embedding (Feature Representation)

- Two technique in word Embedding
  (i) word 2 vec
  (ii) Glove

word Representation

$|V| = 10,000$ (10,000 words)

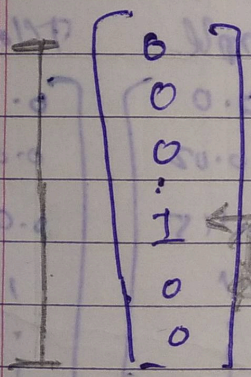
Vocabulary size (Dict) ← dictionary having (10,000) words

- Bag of words (one hot Representation)

eg

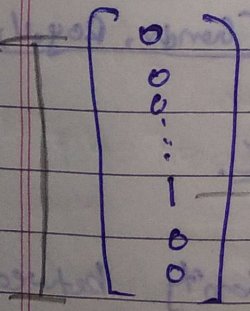
Man [5000] man presentation in 5000 location

If i want to convert man into vector in the form of one hot Representation,



10,000 word representation
other words will become 0
index of 5000 It is difficult to Execute
at index 5000 the value will be 1

woman [9000]



9000 location (present in 9000 location)

→ This is vay sparse matrix, Higher Dimension.

→ Sparse matrix ⇒ is basically means that you have so many number of 0s and you have less number of 1s

* Cosine similarity :- used in [illegible] newely used

PAGE No.
DATE / /

disadvantage :- when my ML or DL algorithm is getting applied, it is very difficult to generalize these vector. we cannot understand like if i want to find out the similer word, it is very very difficult to understand from this, because some of the other index in every word you know that only one vector will be ~~one~~ 1 and remaining all vector will be 0, Hence there is no semantic information of the words.

- to overcome this problem we use word Embedding (feature Representation)

	(2000) Boy	(5000) Girl	(6000) King	(9000) Queen	(1000) Apple	(7000) Mango
Gender	-1	1	-0.92	0.93	0.0	0.1
Royal	0.01	0.02	0.75	0.96	-0.02	0.01
age	0.03	0.02	0.7	0.6	0.95	0.92
food	:	:	:	:	:	:

[300 x 10000] (feature)

→ Converting vector based on some features (Gender, Royal, age)

* Lower Dimention and Dense matrix.

→ Cosine : is used to find similarity between two vectors

PAGE No.
DATE / /

Analogy (Based on Gender)

Boy —represent→ Girl

x_1 x_2 ← x_1 and x_2 are vector of Boy and Girl

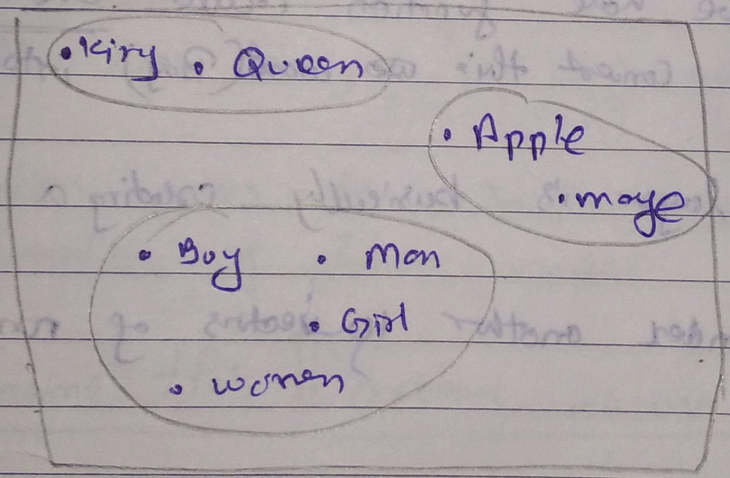
$x_1 - x_2 = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $(-1-(+1)) = -2$

cosine similarity

King —represent→ Queen

$= \begin{bmatrix} \approx -2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ $(-0.92 - (0.93)) = \approx -2$

300D to 2D



Grouping based on similarity

* word Embeding is used to find out the most similar vectors because through the vector it's created in that particular way only

$|V| = 10000$

↑

Vocabulary size tells us size of the Dict

dimension = features [Gender, Royal, age ..]

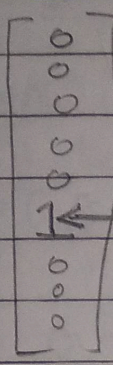
* Bidirectional RNN

Vocabulary → |V| = 10,000

Embedding Layer ⇒ Convert some kind of vector representation

Boy is good

↳ Boy → 2000 ← index



Using Keras) converted Boy into one-hot) After applying one-hot we will get index of these values

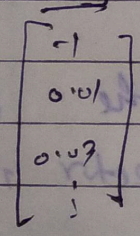
[2000, 4000, 5500] (Boy, is, good)

we will pass this values w.r.t Embedding layer

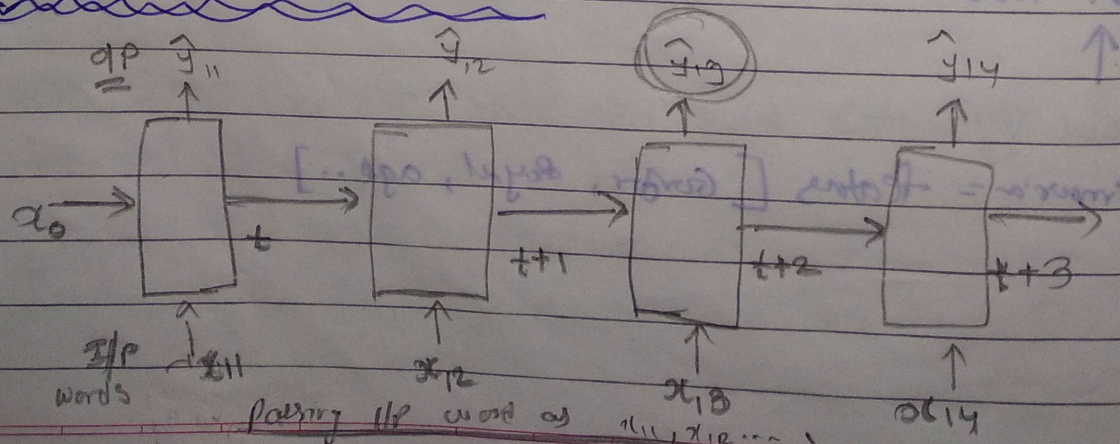
Embedding layer

- In Keras we have function called One-hot, which helps us to convert this word (Boy) into one-hot repres
- Embedding layer is basically creating a feature rep
- for 4000 index another 10 vectors of number will come

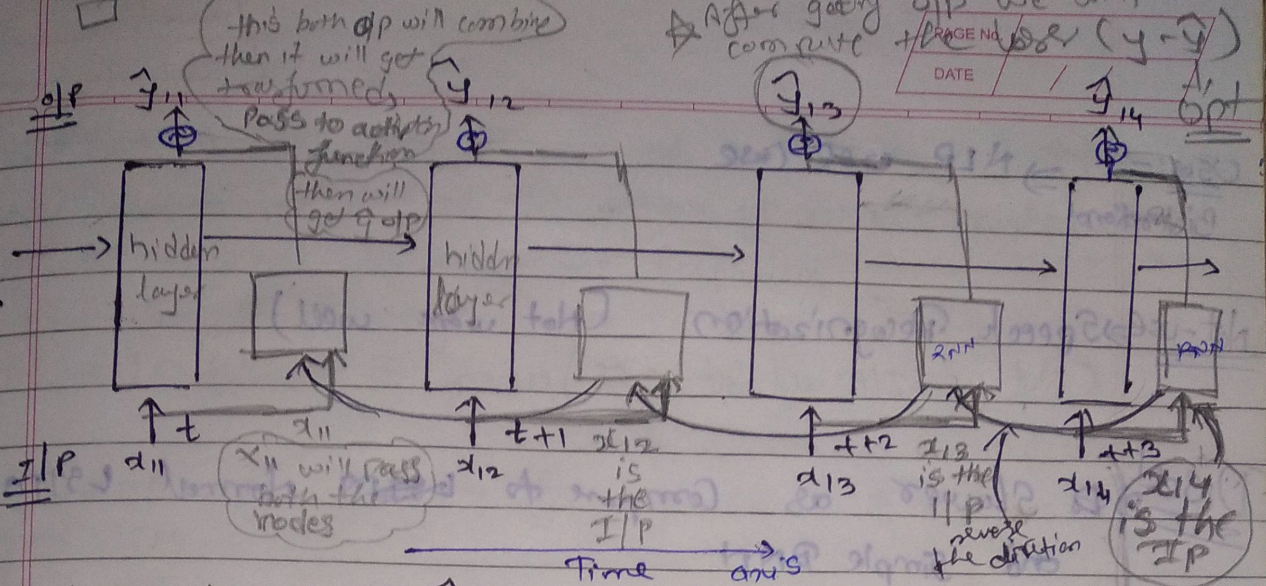
Boy



* Bidirectional RNN



RNN ← Activation function



Explanation ⇒ To Compute o/p ⇒ $\hat{y}_{13}$ we need to have information of x_{13}, x_{12} and x_{11}, This information is getting passed along with the o/p $\hat{y}_{11}$, $\hat{y}_{12}$ ...

Note ⇒ But, what if $\hat{y}_{13}$ this particular o/p is dependent on future words.

eg:- He likes eating ? ... (? ⇒ can be apple, banana ...)
(next word ?)

RNN ⇒ Information passes in Unidirection, it cannot get the information of future words.

- To overcome this ⇑ problem we use Bidirectional.

→ We will be using Backward layer RNN ☐ and it will just change the direction of the information

→ Now, we will just reverse the direction,

Imp → Only thing we are doing is that we are adding the extra LSTM Layer and information is passing to the reverse direction.

Now :- $\hat{y}_{13}$ is able to get information of previous words and the upcoming words

Used in NLP task

DATE

Uses:- ⇒ NLP use case

Bidirectional

Not-use ⇒ Speech Recognization (Not work well)
(Not get I/P all at ones)

Disadv
- Slower as Compare to ~~LSTM~~ Normal LSTM RNN, and Simple RNN
- Because we are taking exact replica of same layer and we are passing information in the bidirectional.

Imp Note:- In the longer sentence, i will get the information of the future words

Pre:- RNN, LSTM, GRU

I am taking o/p after the last RNN layer

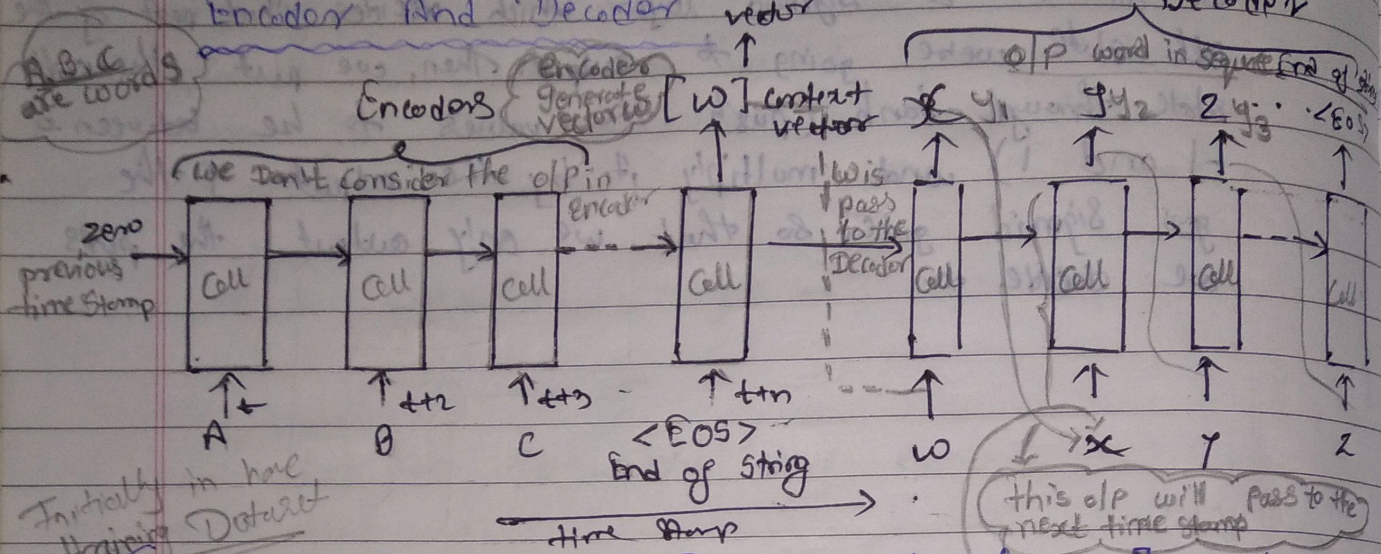
o/p not consider in Encoder we are more focus on getting the context vector

then, context vector will pass to the decoder

PAGE No.
DATE

* Sequence To Sequence Learning with NN

Encoder And Decoder



Initially in our training Dataset

eg:- Language translation [English → French] this is sequence to sequence learning.

eg:- Image Captioning

eg:- Google search [whenever we are searching Dog image we will get images on Dog]

eg:- LinkedIn [we get some automated reply in the chatbox]

* Two main Component in Sequence to Sequence

1. Encoder
2. Decoder

- Encoder ⇒ (Encoder it can be any NN like RNN, LSTM, RNN) But usually we mostly select LSTM because it works well

⇒ In Encoder the o/p will not be present the o/p will present in the last time stamp <EOS>

* Encoder and decoder are in unidirectional

One hot (w o r ds) wf 1 0 0 0 0 (Cell can be any NN)

PAGE No.
DATE / /

eg • In language translation we pass text, each and every text is passed in the vectors

Vectors like ⇒ { Embedding layer
Onhot Representation
Word2vec

process • Convert the word to vector then pass to NN

$x = \langle x_1, x_2, x_3 \rangle$ .... It can be any no. of i/p we consider as A, B, C

$y = \langle y_1, y_2, y_3 \rangle$ .... Output,

* • for prediction $\hat{y}_1, \hat{y}_2, \hat{y}_3$ (Training Data-Set)
LSTM works on the probability

probability

$Pr(\hat{y} \mid \langle x_1, x_2, x_3 \ldots x_n \rangle)$

{ Sequence of i/p will be taken and we will find the probability of $\hat{y}$.

• $\hat{y}$ is the combination of ⇒ $\hat{y} \langle \hat{y}_1, \hat{y}_2, \hat{y}_3 \rangle$

• After getting $\hat{y}$, then i will try to find out the loss $(y, \hat{y})$, for this i will use Optimizer.

Adam ⇒ Optimizer will reduce the loss function, through the Backpropagation, we gets updated weights.

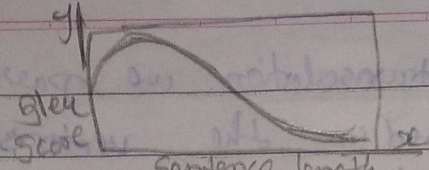
Disadvantage ⇒ { As the sentence length increase w.r.t the Encoder and decoder, the accuracy decreases. (Sentence length eg- 100 words)

To avoid this problem we will be using Attension

Cell = RNN

PAGE No.
DATE / /

Bleu Score

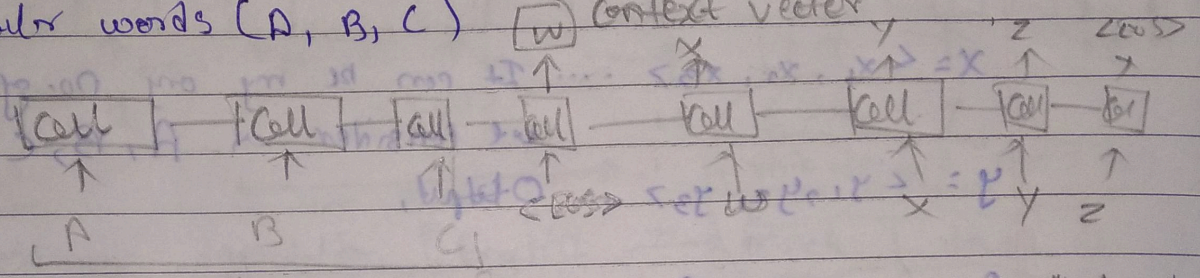


Imp

→ As the sentence length increasing, the BLEU score (decreasing) coming down, hence this was the problem researchers found

Disadv

Note:- whenever the information is long this vector [w] generated that is actually is not able to capture all the information of these particular words (A, B, C)



* Attention Model

→ Initially we will just take the window size words in a sentences may be 3 or 4 depends on hyperparameter that we select.

→ If i select 4 to 5 words might in the sentences then ill pass that particular information to the decoder, the decoder will convert it it goes on like this.

Hence, to solve the problem of Encoder and Decoder this is the solution it's called as attention model.

Disadv Encoder Decoder

① Encoder and Decoder usually do not perform well longer sentences

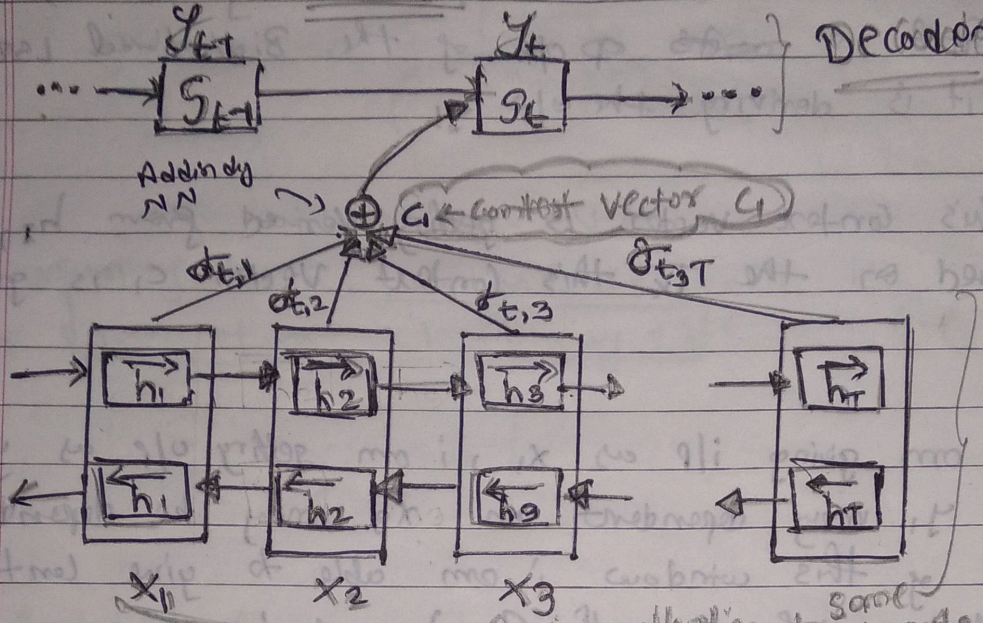
② Researchers experimented and found out lower Bleu Score

DATE

* Understanding Attention Models and Architecture

Refer => RNN, LSTM, Bidirectional LSTM, Sequence 2 seq

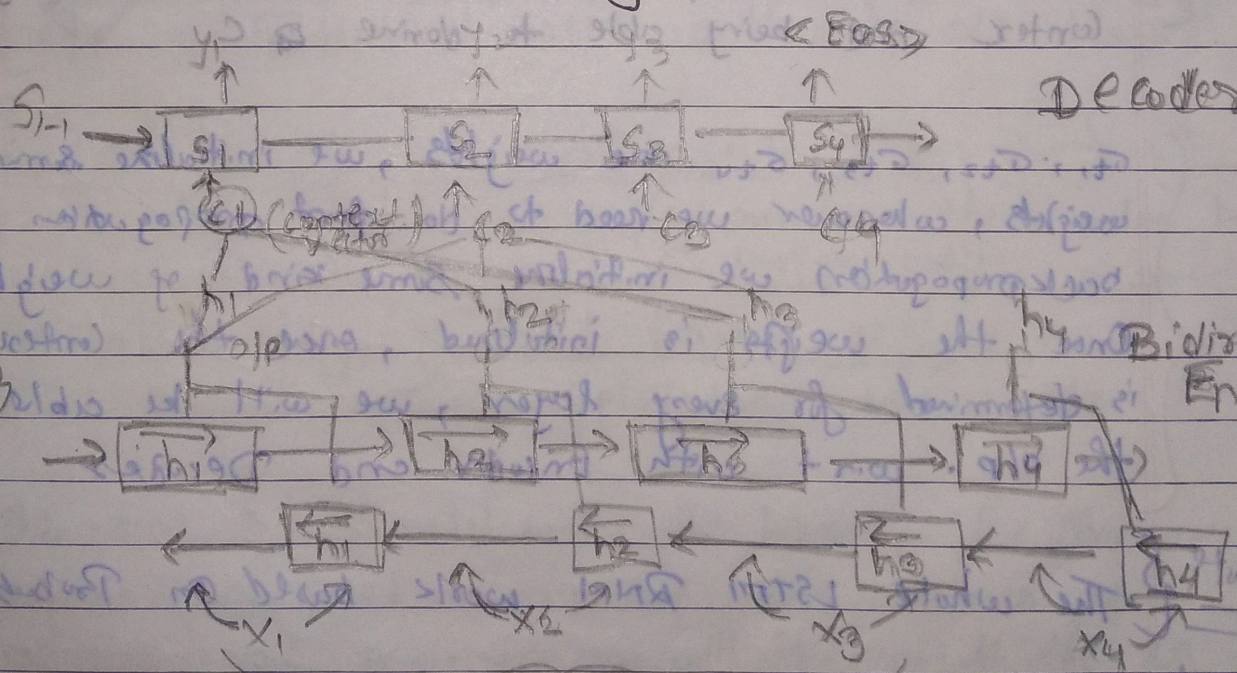
LSTM RNN



$\alpha_{t,1}$ $\alpha_{t,2}$ $\alpha_{t,3}$ $\alpha_{t,T}$ are basically weights

giving attention to same words based on Tx parameter

In encoder we will be using Bidirectional RNN
In Decoder we will be using LSTM RNN



S_1, S_2, S_3, S_4 states of Decoder

$h_1, h_2 \ldots$ o/p

we have to consider this window, and this window is represented as Tx, here Tx = 3

This window can be decided as a hyperparameter. like initially how many words should i give, then the algorithm will able to translate

Eg → Language Translation
French → to → English

PAGE No.
DATE / /

Read Property

Explanation

→ Context vector are getting derived from h_1, h_2, h_3 (taking window size = 3 # of T_x)

Now this h_1 will combine this, since my $T_x = 3$, so my first 3 ~~words~~ of o/p of the Bidirectional LSTM and it is deriving the o/p y_1

This context vector C_1 is getting derived from h_1, h_2, h_3 so, Based on the T_x this context vector C_1 is getting derived

When i am giving i/p as x_1, i am getting o/p as y_1 where, y_1 may dependent on x_3 may be dependent on x_2 Because of this window i am able to give context ~~h2 and h3~~ information through h_2 and h_3

Even this y_1 is dependent on x_2 and x_3.. it is getting context and being able to derive C

$\sigma_{t_1}, \sigma_{t_2}, \sigma_{t_3}, \sigma_{t_0}$ are weights, we initialize some kind of weights, whenever we need to do front propagation and backpropagation we initialize some kind of weights. Once the weights is initialized, once the context vector is determined for every layer, we will be able to see the o/p. work with Encoder and Decoder

Note → The whole LSTM RNN work based on Probability

Advantage → Language Translation
→ Text Summarization
→ Q, A

Encoder is → Bidirectional
Decoder is → unidirectional

PAGE No.
DATE / /

Maths

- **Condition 1**

when ⇒ T = 3

$\alpha_1, \alpha_2, \alpha_3$

α_1

(T)

window size = 3, i'll be getting 3 α values for the first word x_1, and this i'll be passing to my decoder and it will create an o/p y_1

- Sum of the α_1, α_2 and α_3 will be,

Imp: Sum of $\alpha_i = 1$ (Always 1) ← (1)

$$\sum_{i=1}^{T_x} \alpha_i = 1$$

- **Condition 2**

Imp: C_t → Context vector, Context vector is generated, how it is getting generated, it is just multiplying, o/p of Bidirectional LSTM with the α values

$$\sum_{i=1}^{T_x} \alpha_i h_i$$

we can change the α value based on backpropagation

- This is the property of **Feed forward NN**
  It is Basically multiply, I/p features and weights

- The feed forward NN is it is basically for Backpropagation because, we need to update the values of α_1, α_2 and α_3. If we will not update the so we will not get good accuracy

eg: Some like ANN (feed forward NN)

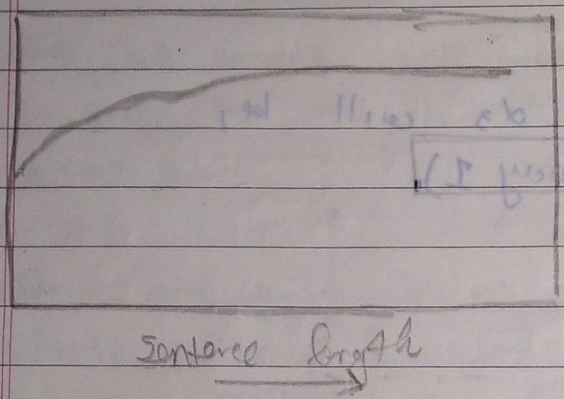
~~Why we use this type of model.~~

PAGE No.
DATE / /

Note: Softmax helps to make α value $\boxed{\alpha = 1}$ always for any long window size, it will make $\alpha = 1$ always.

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad \cdots e_{ij} = a(s_{i-1}, h_j)$$

↑ feed forward NN



even the sentences are long we are good accuracy!

Eg: In Google Translator, even for longer sentences, it is able to give good accuracy. Because of the Time Stamp probably, time stamp acting as the parameter the sentences are changing.

Q Why we use this type of Model?

⇒ In case of Encoder and Decoder, when the researcher were tried with different - different sentences length, does not perform well with the longer sentences.

Because of this they have included Bidirectional entire LSTM in decoder, they have made some architecture changes between basically in the Context they have use Attention model which is nothing but feed forward NN w.r.t some Time stamp which is basically indicating the window size.

PAGE No.
DATE

Encoder, Decoder, Attention Models, Transformers, Bert - I

CNN always try to train in way to understand a relationship of an static data, but it will not be able to memorize, it will not be able to understand a context, if it is not able to understand a context, eg if i am talking about a Question system you will not be able to build it. even like a sentimental system you will not be able to build it. Language translation you will not be able to build it, text summarization you will not be able to do it.

To overcome this problem CRNN) → introduce LSTM

- Seq to seq
- vec to seq
- seq to vec
- vec to ve

Note :- ○ ← it can be any type of RNN cell

① ○ o/p — used in = Image classification, Bounding box regression
↑
○ h
↑
○ I/p

I/P → I O/P

f(x) regression (approx)

vector → vector

x → ŷ ... we go from 1 vector x to another vector ŷ

cell can be RNN, LSTM, GRU

② which then predict seq of o/p
↑
sequence of internal state

○ ○ ○ o/p
↑ ↑ ↑
○ → ○ → ○ h
↑
○ I/p (single I/P)

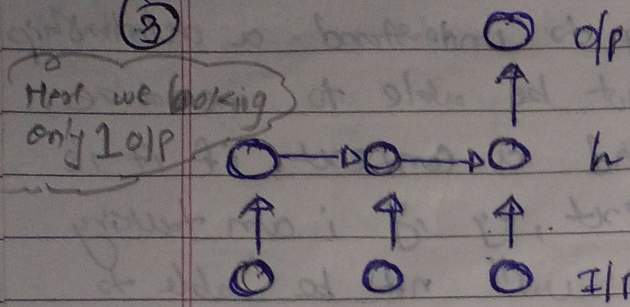
used in :- Image captioning

x → ŷ[t]

eg:- feed back

vector → sequence

③



Used in = Equation Solver
program Executer
Sentiment Analysis

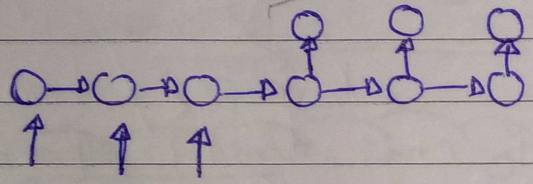
See I/p a seq and we Expect a [vector o/p

$x[t] \rightarrow \hat{y}[T]$

Sequence → Vector

Sequence of I/p feed into the Hidden layer then we produce an o/p only at the End

④



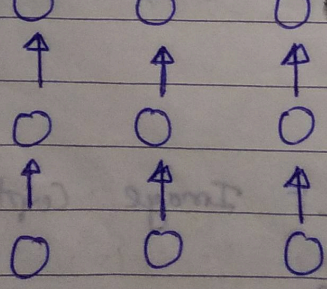
⇒ Neural machine Translation

$x[t] \rightarrow h \rightarrow \hat{y}[t]$

Seq → vec → Seq

Seq to Seq, but we also came though a condensed representation (which) is basically a vector, so in this case we can call this scheme also encoder decoder

⑤



Used in = Autocorrection

$x[t] \rightarrow \hat{y}[t]$

* Seq to Seq

Issue in Encoder

PAGE No.
DATE / /

* Separated I/P and O/P Network ~~with neural~~ as

- I/P Network Named as ⟶ ENCODER
- O/P Network Named as ⟶ Decoder

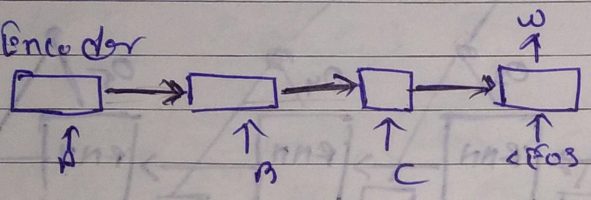
Context vector (w)

- Context vector ⟹ we are trying to give multiple I/P (eg: I am fine) it is trying to understand, it is trying to summarize Each and Every word and then it is trying to send this (w) input to a different Network Giving I/P from the previous network → decoder (O/P Network) → Encoder

Issue in Encoder

Note

In Encoder



→ In Encoder we are trying to propagate our data only in one direction.

→ The word that you are talking before and the word that you are talking after that matters a lot.

But in Encoder the my dataset is propagating only in one direction.

$$h_t = sigm\left(w^{hx} x_t + w^{hh} h_{t-1}\right)$$

h_t → It is a kind of O/P that you accept from a particular cell

x_t → current I/P

h_{t-1} → feedback from previous

$$y_t = w^{yh} h_t$$

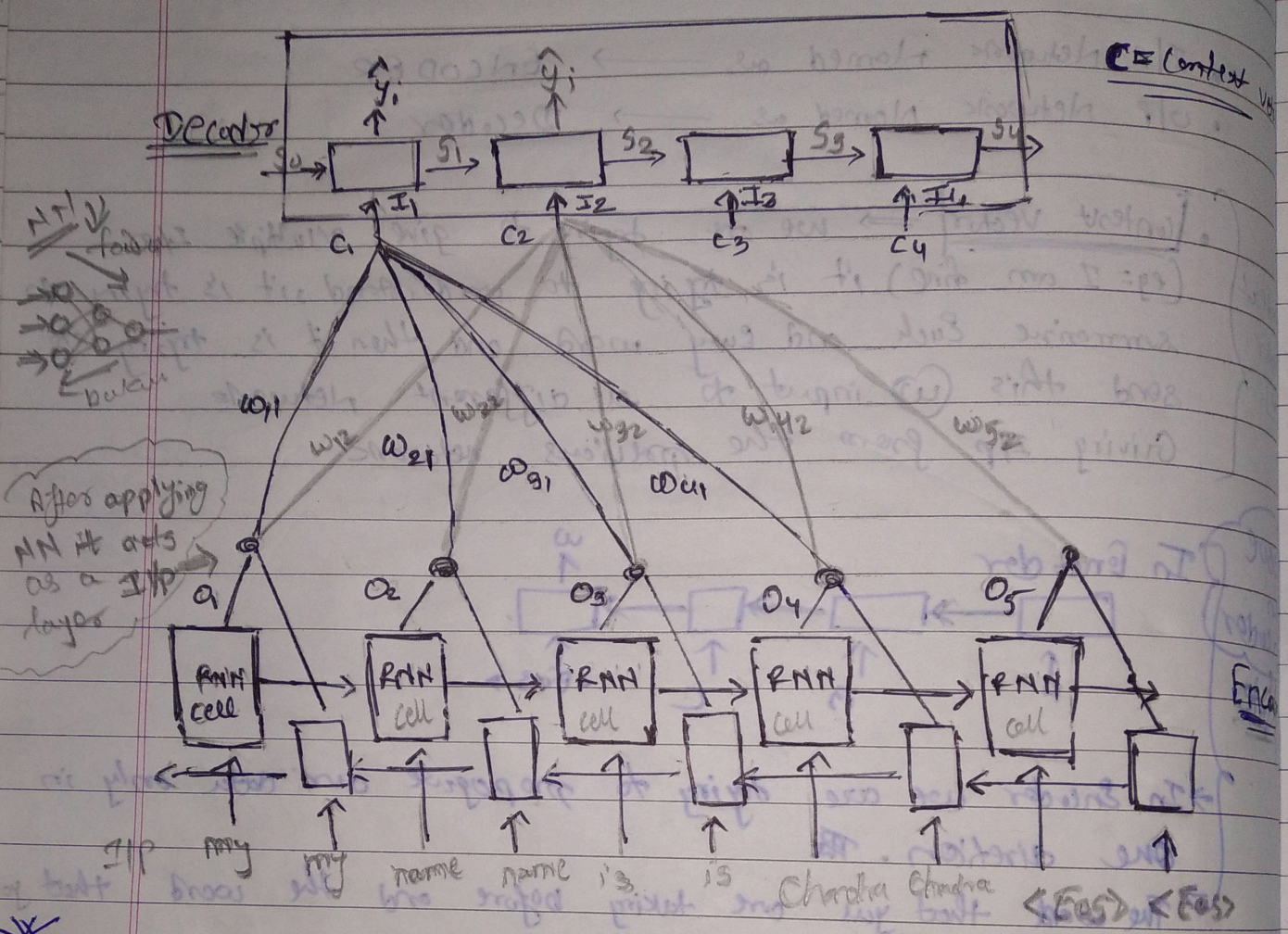
this formula used in

* → Internal hidden layer calculation

$$P(y_1, \ldots, y_T \mid x_1, \ldots, x_T) = \prod_{t=1}^{T'} P(y_t \mid y_1, \ldots, y_{t-1})$$

after Naive Bias

Attention Models, Transformers - Part 2



Note

In between we introduce NN to give attention for some of words.

eg:- Bull is gaining high

$O_1, O_2, O_3, O_4, O_5 \Rightarrow$ O/P of Bidirectional

$w_{11}, w_{12}, w_{21}, w_{41} \Rightarrow$ weights (weights will be random)

Explanation:

After Introducing NN in between the Encoder and Decoder
The O/P of Encoder (Bidirectional) will acts as I/P.
then after apply weights (i.e, $w_{11}, w_{21}, w_{31}, w_{41}$)
Then it is given to I/P of Decoder, after that we get O/P predicted ($\hat{y}_i$) O/P.

DATE / /

$$\Rightarrow \boxed{C_i = \sum_{j=1}^{Tx} \alpha_{ij} h_j}$$

where,

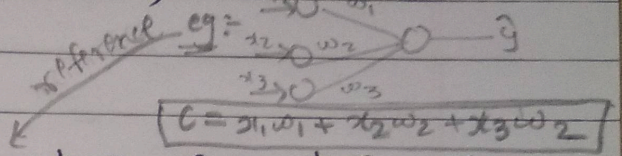
$C_i \leftarrow$ {Context Vector I/P which will get inside a decoder

$C_i = I_i$, Tx = hyper meter (no. of I/P into my NN)

$h_j = O_i$ (output) combined output of RNN network

α = weights (ω) (learnable parameter)

$\alpha_{ij} = \omega_{ij}$



Context Vector

$\Rightarrow \therefore C_i = I_1 \Rightarrow O_1 \omega_{11} + O_2 \omega_{21} + O_3 \omega_{31} + O_4 \omega_{41}$

Calculating weight $\Rightarrow$ Exponent

$$\alpha_{ij} = \frac{Exp(e_{ij})}{\sum_{k=1}^{Tx} exp(e_{ik})} \quad \cdots\cdots \text{Calculating weights}$$

$ik \rightarrow h_1, h_2, h_3, h_4, h_5$

Attention

$$e_{ij} = a(s_{i-1}, h_j) \quad \cdots\cdots \text{(Attention Equation)}$$

↑ attention function

s = feedback

s_{i-1} = the o/p that we are getting from previous one, s_i s_1 (as I/P (or) o/p of decoder

h_j = o/p from the Encoder

Condition

$\sum \alpha_{ij} = 1$ (always +ve) — (1)

$\alpha_{ij} \geq 0$ (α_{ij} will be always +ve)

3000 × 15 = 45000 0

Application

(i) Language Translation
(ii) Q A System
(iii) Speech Recognition
(iv) Image Captioning
(v) Text Summarization