RNN

Sequential Data

ANN → tabular data

CNN → Images

RNN → Recurrent NN

↳ is type of Sequential model to work on sequential data.

Non-Sequential Data ↴

iq | marks | gender | Placement

Sequence doesn't matter if we change iq ⇄ marks

Sequential Data

text

(i) Hi my name is Dhanraj.

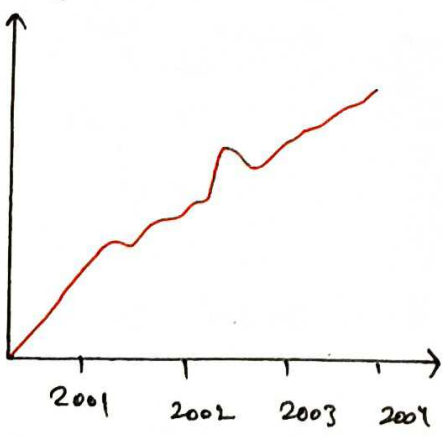
↳ Here we cannot send data like "Hi name my is Dhanraj"

So, Sequence matter

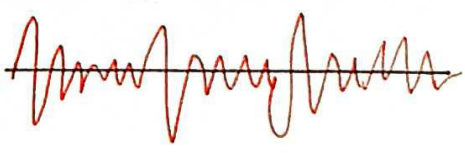
(ii) Time Series Data

(iii) Speech Data / Audio

Time:



Speech:

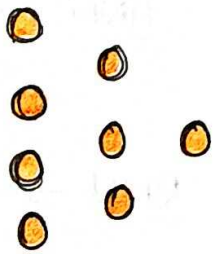


RNN → NLP CNN → images → Computer Vision

Why use RNN?

Input	Output
Hi my name is Dhanraj (5)	0
I Love cars (3)	0
India Won the match (4)	1

ANN → text classification

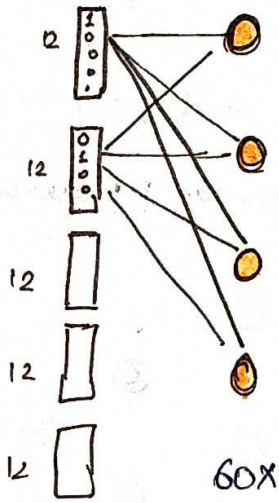


→ Find unique words in data
Total unique words → 12

→ Vectorize (OHE)

[1, 0, 0, 0, 0, ...] → Hi
↳ 12 items

[0, 1, 0, 0, 0, 0 ---] → My
↳ 12 items



60 × 4 = 240 Weights

According to first line input size is → 5
Acc. to second line input size is → 3
Acc to Third line input size is → 4

Problem :
This means that no of input change in every line. But we cannot change No. of Input ANN for every line. No. of input is fixed.

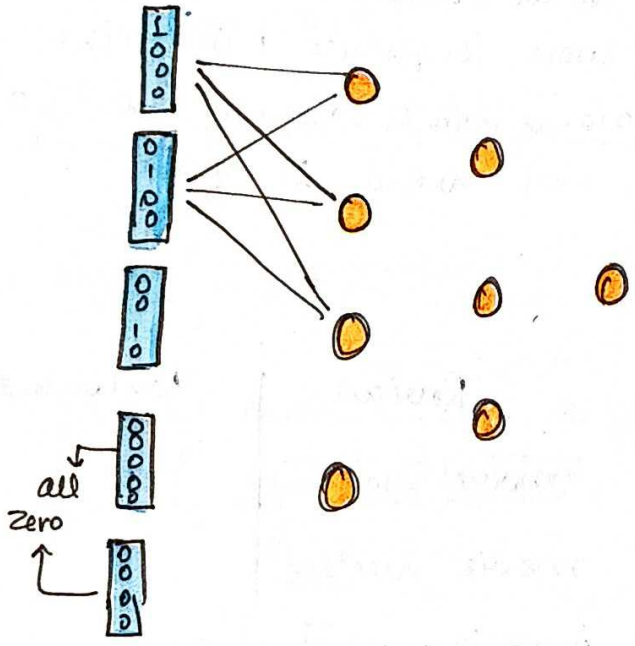
Solution → Use Zero Padding

→ First, we have find maximum word of line.
In our case → last line (Max word (5))

→ Create ANN → Input with (5) ↳ No. of input.

→ create a padding (zero padding) for less word input line.

for eg:- 2nd line contain (3 word)

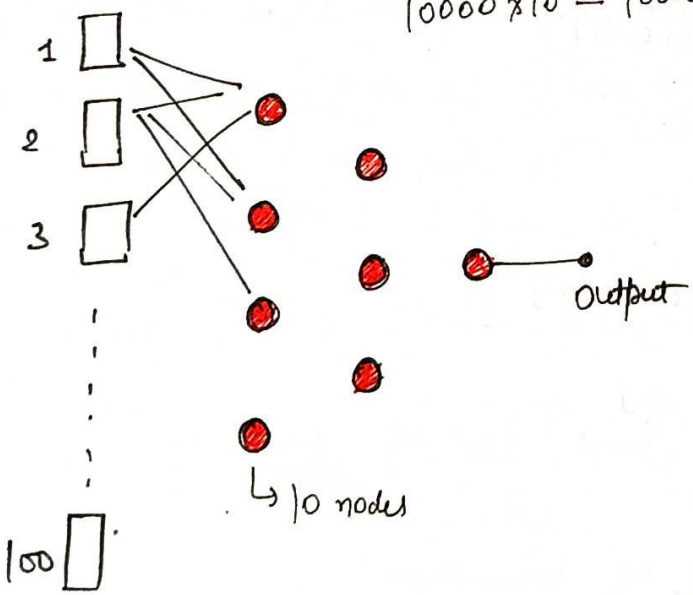


* But this is not a perfect solution. We can say "jugadu" method to solve these type of problem explaing]

Let say we have
Vocab → 10K words
And max words → 100
and min word → 5.

← So, when we create ANN and No. of Input for min word is 5 and 95 is Zero padding. At this point thi is irrelevent

10000 x 10 = 100000 weights



↳ Last 95 input is zero padding

* Zero padding fail → When Input size is = 100 words and during prediction time someone enter → 200 words At this moment zero padding fail.

Problems :-

1. text input → varying size
2. Zero padding → Unnecessary computation
3. Prediction problem
4. Totally disregarding the sequence information. (We take a sentence and send in neural network but we loose sequence info. like (My word after Hi name word after My etc.) or Hi word is first or my word is first.

Data for RNN

(timesteps, input-feature)

Vocab → 5 word

movie [10000] was [01000]

good [00100] bad [00010] not [00001]

Review	Sentiments
movie was good	1
movie was bad	0
movie was not sad	0

Review 1

[[10000], [01000], [00100]]

(3, 5) → No. of features

↓ no. of timesteps

t=1 → Send one word

t=2 → Send two word

t=3 → Send Three word

review 2 → (3,5) and review 3 → (4,5)

Keras → SimpleRNN → batch size, timesteps, input features

(3,4,5) → 3-D tensor

How RNN Works?

Review (X)	Sentiment
x_{11} movie x_{12} was x_{13} good → X_1	1
x_{21} movie x_{22} was x_{23} bad → X_2	0
x_{31} movie x_{32} was x_{33} not x_{34} good → X_3	0

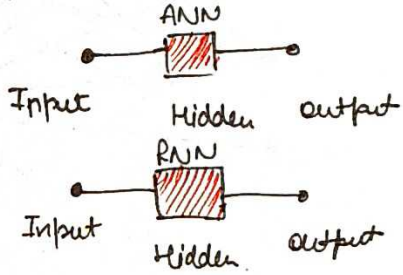
movie [1 0 0 0 0] was [0 1 0 0 0] good [0 0 1 0 0] bad [0 0 0 1 0] not [0 0 0 1]

RNN ↔ ANN

↓

2 big diff

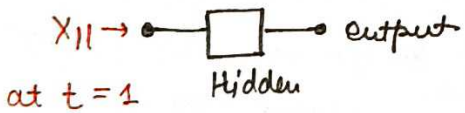
* Structure of RNN and ANN is same.



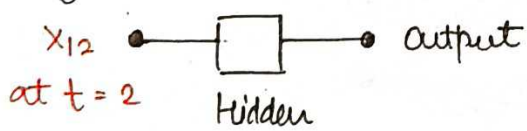
→ First different is input.

In RNN input → on time basis

* X_{11} goes in input at time t_1.



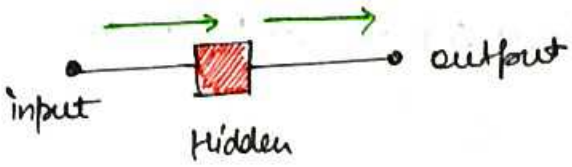
* X_{12} goes in input at time t_2.



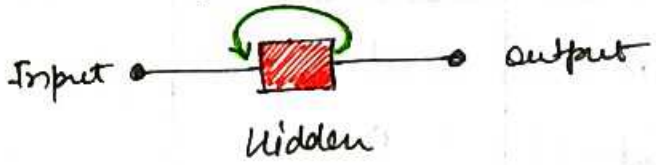
So, In RNN we cannot send all word at a time. Send word one by one on the basis of time.

→ Second difference is feed back

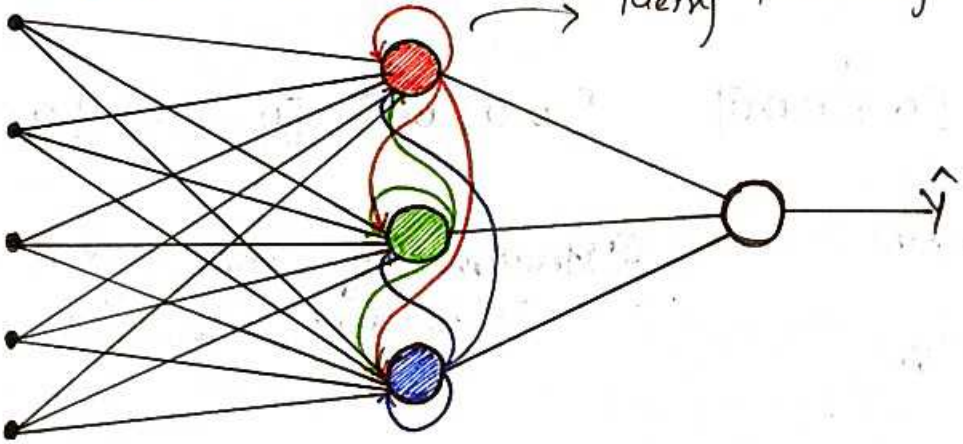
In ANN information send from input to output.



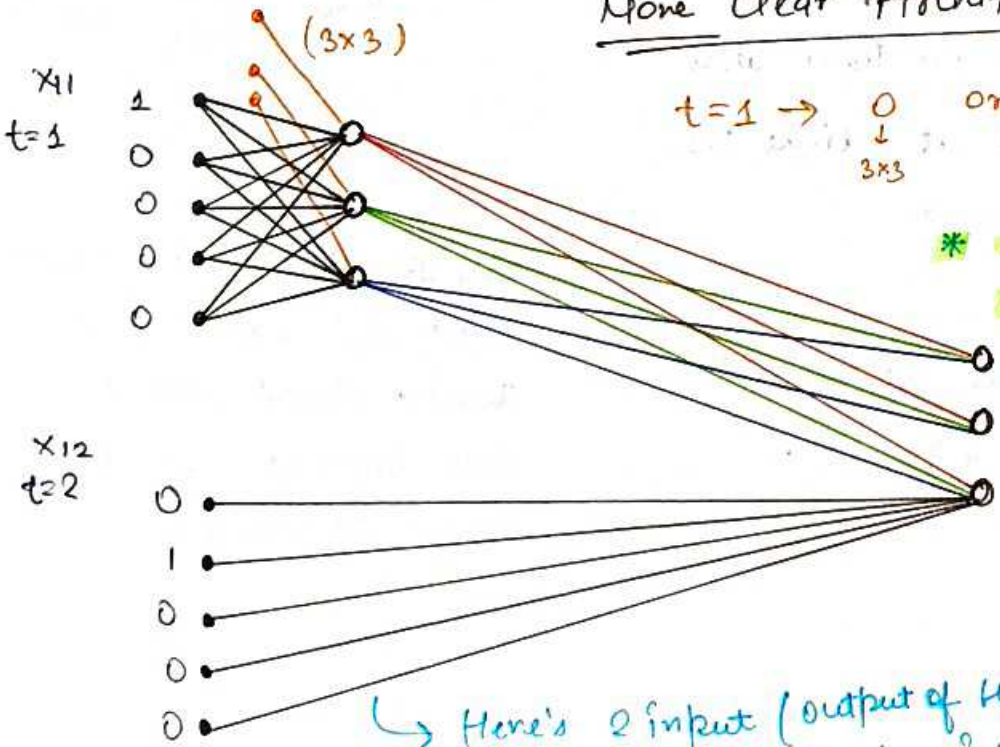
but In RNN information return as the feed back



Architecture:



More Clear Architecture

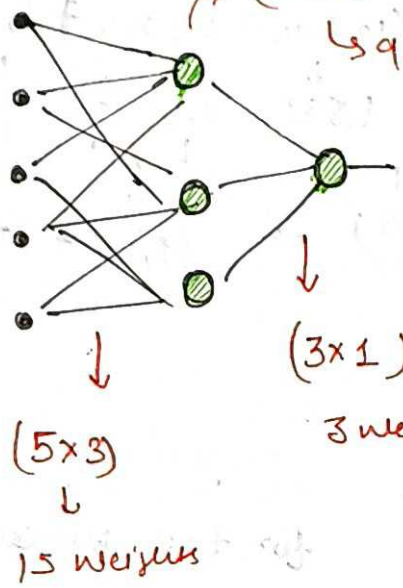


$t=1 \rightarrow$ 0 (↓ 3×3) or random (↓ 3×3)

* at t=1 also want 2 input. That's why use 0 or random number because at t=2 → 2 input

↳ Here's 2 input (output of Hidden layer of t=1 and input → X12)

Weights and bias →



(3×3) → Hidden Output → Input Hidden
↳ 9 weights

(3×1)
3 weights

(5×3)
↳
15 weights

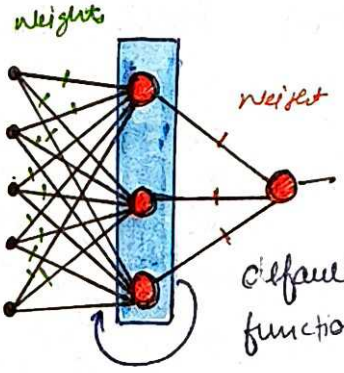
Total weights = 15+9+3 = 27

Total bias = 3+1 = 4

27 Weights and 4 bias

RNN Forward Propagation

Review				Sentiment
X_{11}	X_{12}	X_{13}		1
X_{21}	X_{22}	X_{23}		0
X_{31}	X_{32}	X_{33}	X_{34}	0



default activation function = tanh

X_{11} → Vectors 5 dim
[1 0000]
↓
one by one

forward → unfolding through time (loop)

t=1 → O_0 → W_i → (3,5)

t=1 → [0000]

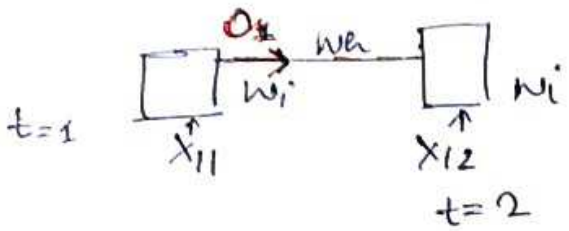
$[X_{11} W_i + O_0 W_h]$ ← X_{11} ↳ (1,5)

(i) $X_{11} W_i$ = (1,5)(5,3) + bias
= (1,3) matrix

(ii) (1,3) Max → $f(X_{11} W_i + b_1)$ into activation = tanh

(iii) After Apply activation function we get O_1 = (1,3) + bias

t=2



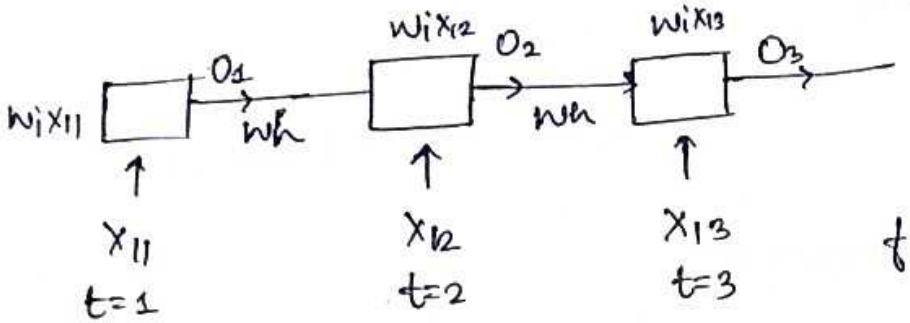
for $t=2$ we calculate

$$f(X_{12} W_i + O_1 W_h + b_1)$$

(1,5) (5,3) (1,3) (3,3)

(1,3) + (1,3) + (3,)

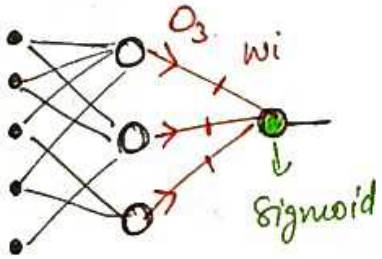
$t=3$



for $t=3$ we calculate

$$f(X_{13} W_i + O_2 W_h + b_1)$$

So, Now first line is completed..



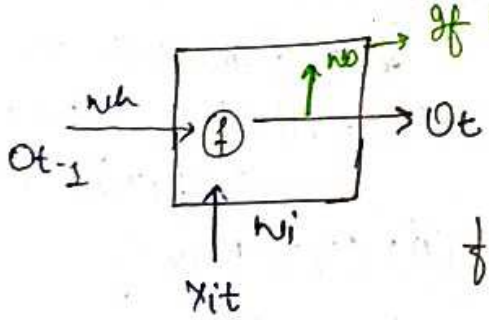
$O_3 W_i \longrightarrow$ Sigmoid

(1,3) (3,1)

(1,1)

Simplified Representation :

$(O_t W_O) \longrightarrow$ activation $= \hat{Y}$



O_t of last timestep

$$f(X_{it} W_i + O_{t-1} W_h) \longrightarrow \text{Vector}$$

i = Row No.

t = timestep

RNN Sentiment Analysis

2 Techniques

- Integer encoding
- embeddings

text	res
Hi there	1
How are you	0

Integer encoding

1. Vocab →

Hi (1) there (2) How (3) are (4) Yoo (5)

2. Replace sentence to their index value.

[1, 2]

[3, 4, 5]

3) Padding → Padding with 0. So, every vector will be same size.

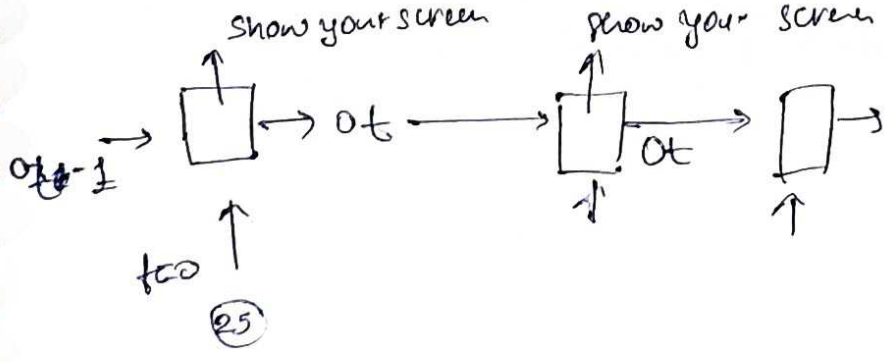
[1, 2, 0] → Padding

[3, 4, 5]

4) Train through RNN.

code → integer_encoding - simple_RNN.ipynb

return_sequence = False



when it is false then not show your screen and show at last one o_t.

Embedding

In NLP, word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.

Example

this is nice

↓

[0.7, 0.1, 0.3]

→ In embedding → In vectors are non-zero value.

→ Semantic Meaning
↳ word kis context mai use hua hai → embedding know.

→ Word 2 Vec / Glove

→ In integer encoding most of the value in vector are zero for eg:- Sentiment

review 1 → 20
review 2 → 2000

So, in review 1 ↳ 1980 value are zero.

Deep Learning → Embedding Learn → Embedding layer in Keras

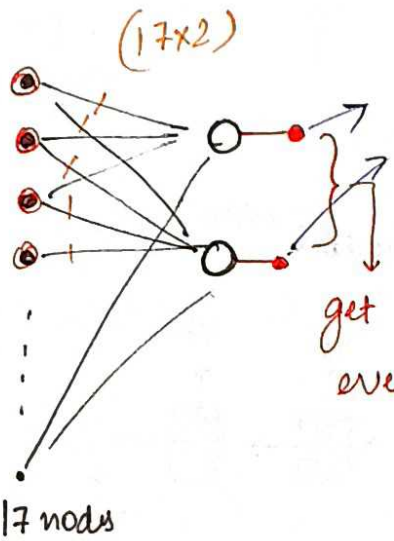
Embedding layer
↓ dense represent
RNN
↓
Sentiment.

Code Architecture

review sentence in number

[9 1 0 0 0] → Total Word = 17

(17,1) dim OHE

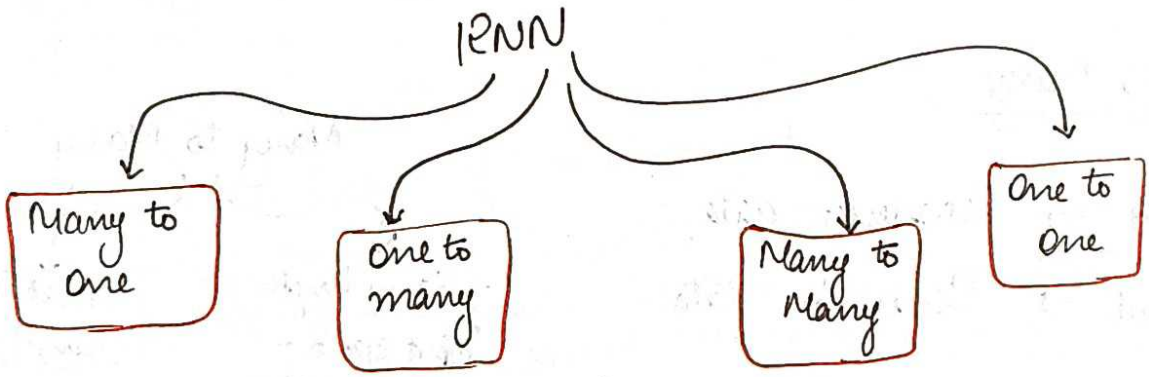


get 2 numbers for every word.

After send one review sentence

[[x1 x2], [x3 x4], [x5 x6], [x7 x8] [x9 x10]]

Types of RNN



Many to One

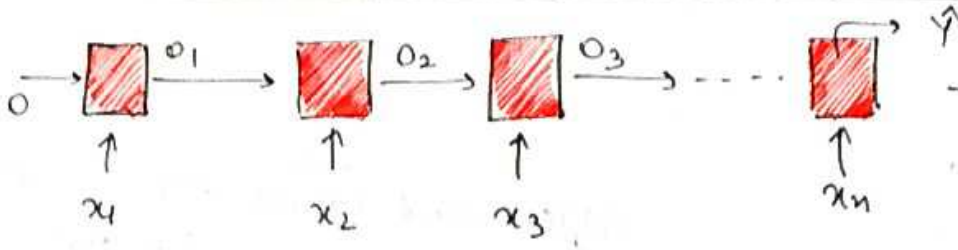
Input → Sequence → sentence, character, time series

Output → Non-Sequence → int/num → scalar → (1,0)

example → Sentiment Analysis

→ Rating Prediction

→ rate movies



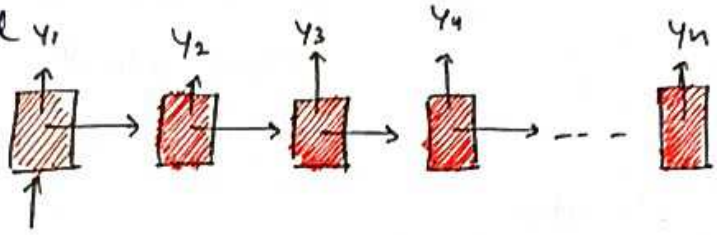
→ Many to One Architecture
Multiple input $(x_1, x_2, \ldots x_n)$
Single output $(\hat{y})$

One to Many

Input → Normal Non-Sequential

Output → Sequential

Architecture :-

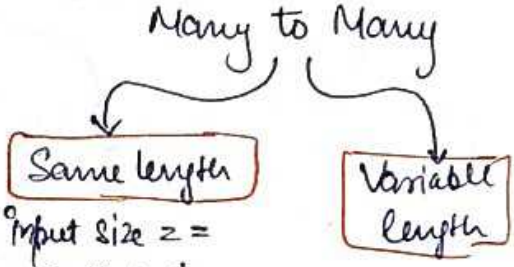


Example → Image Captioning
↳ Music generation

Many to Many

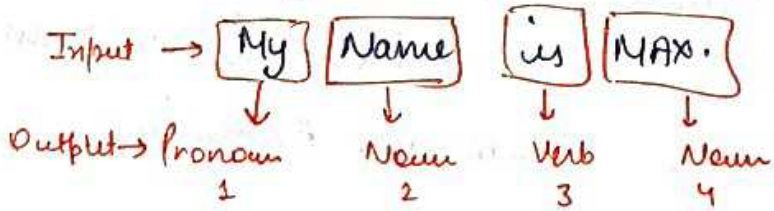
Input → Sequence data

Output → Sequence data

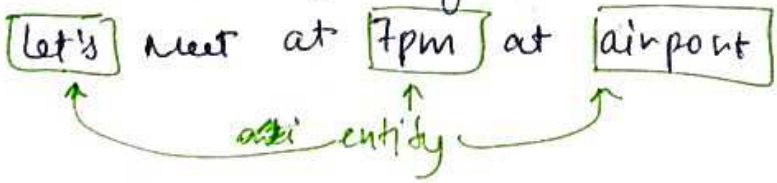


example: ←

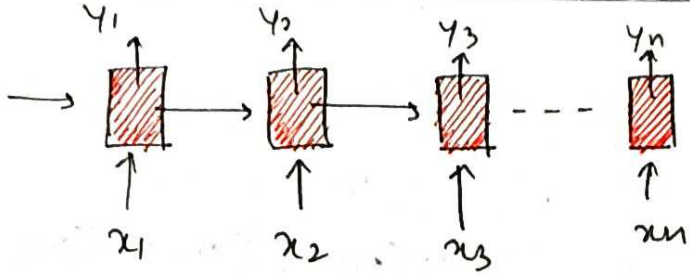
(i) parts of speech tagging in NLP.



(ii) Name Entity Recognition → Chatbots



Same Length Many to Many Architecture



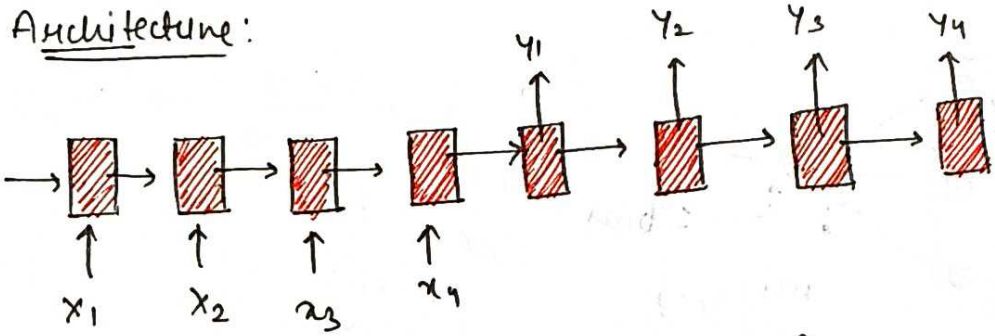
Variable Length Application :-

Machine Translation → google Translation.

Hi my name is Max.

मेरा नाम मक्स है।

Architecture:



Why design this type of architecture?

→ Because we cannot translation sentence → word by word we have to know whole sentence then we start translate. Same goes with this architecture. First read all word then start Translate.

One to One

Input → non-Sequence data

Output → Non-Sequence data

Application :- Image Classification

Architecture!

* Technically One to One RNN → Simple NN (ANN)