DATE :__/__/__
PAGE :_____

~~Image formation~~

NLP
Natural Language Processing

* **Applications**
  - → Email → autocompletion
  - → Google translate
  - → Spell checker
  - → Question answering (ChatGPT)
  - → Text from image
  - → Text summarization

↓

RNN
↓
Recurrent Neural Network

(Base layer)

→ LSTM
→ GRU

stores output of previous layers

we have to store previous content.

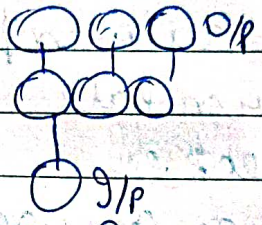
DATE :__/__/__
PAGE :

Seq to Seq Model

Many to one

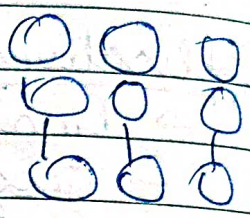
eg sentiment analysis

One to many



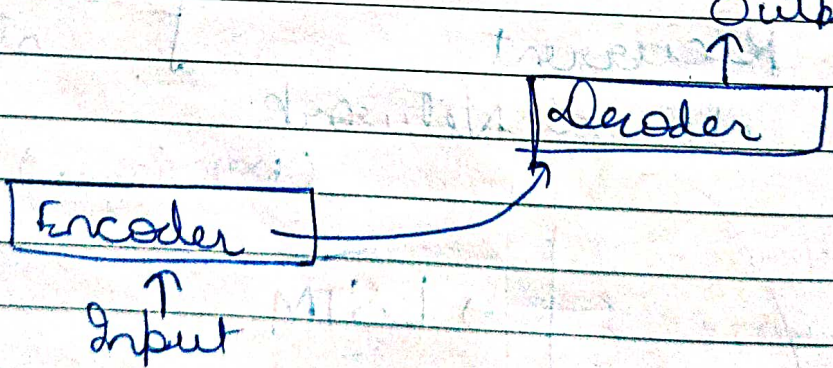
eg - Synonym generator

Many to many



eg language translation

Encoder Decoder architecture (RNN)



Text → Vector

↓

Mapping is done

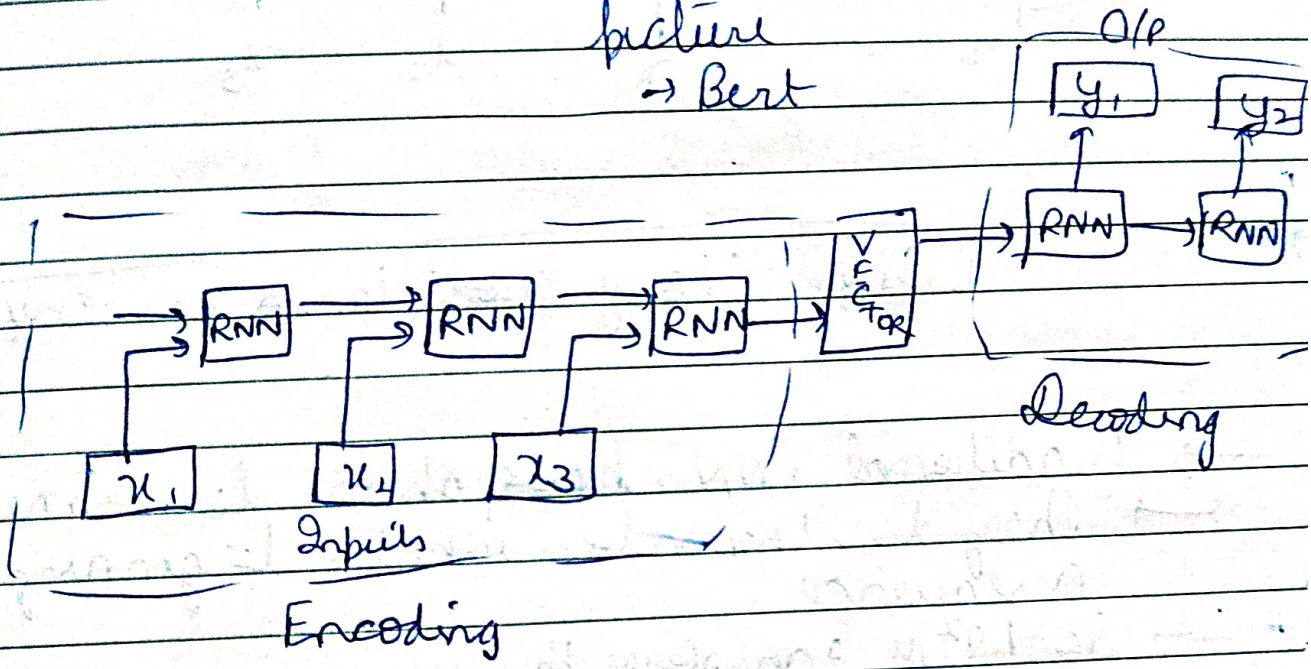
DATE :__/__/__
PAGE :_____

Disadvantage (Vanishing gradient)

↳ Can't decode large sequences/sentences.

↓

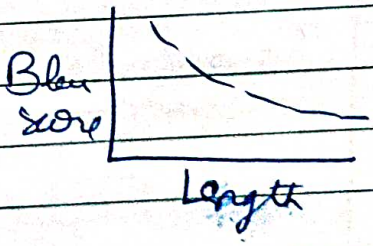
Attention based model came into picture → Bert



Bleu Score (Accuracy concept in RNN)

↓

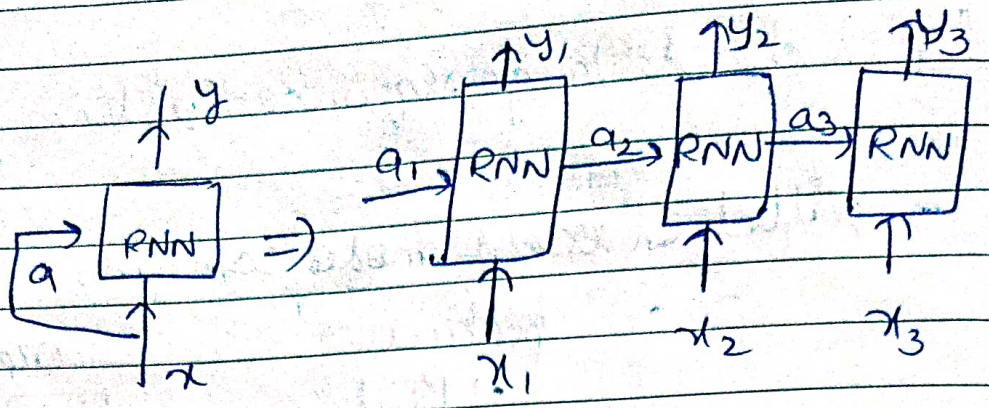
Length $\propto \frac{1}{\text{BleuScore}}$



Why not ANN?

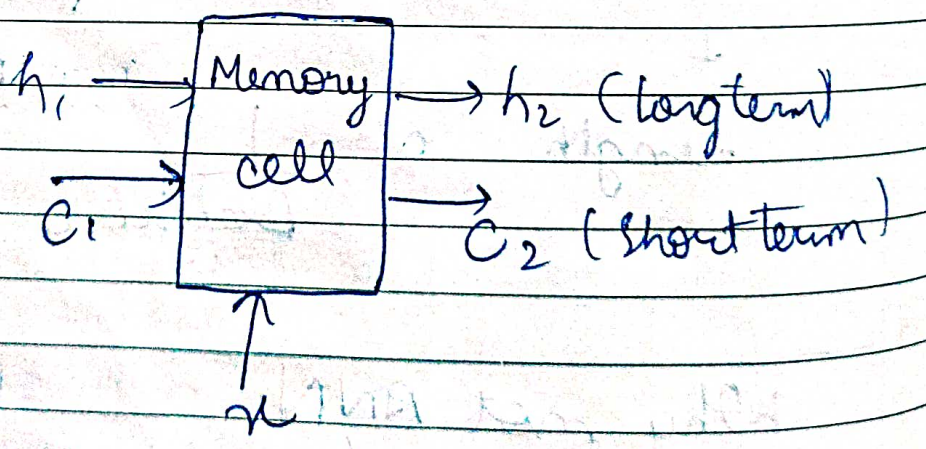
↳ Cause sequence is v. v. imp in Textual data

RNN - basic model

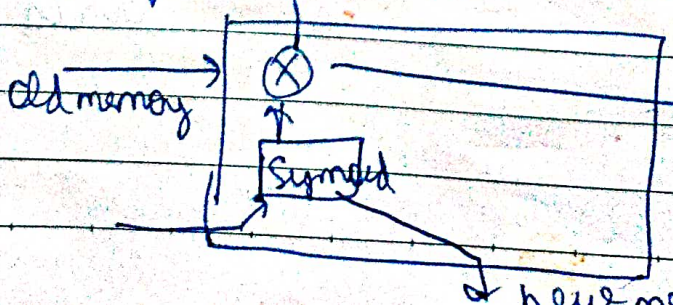


LSTM (Long Short term memory)

→ Traditional RNNs have short term memory
→ They don't remember what's beginning of a sequence
→ So LSTM can solve this issue.



Forget gate (Discard the previous memory)



DATE : __/__/__
PAGE : _____

Eg TEXT GENERATION

eg I love reading _ _ _ _ _ _

↳ Let the model predict the output

eg output → book, novel, newspaper etc

$\hat{y}$ (target)

This can be done by CONDITIONAL PROBABILITY

$$P(A/B) = \frac{P(B/A) * P(A)}{P(B)}$$

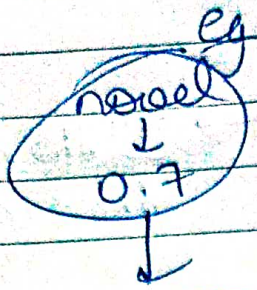
↳ P(A) given B has already happened.

Input (Tokens)	Output (Target)
(I) x_1	$\hat{y}$
(love) x_2	
(reading) x_3	(novel, book etc)

DATE : __/__/__
PAGE :

$P(\hat{y}/(x_1, x_2, x_3))$

One which has most probability



will be chosen

Smilary : it continues to predict upcoming words

Processing

→ Import the dataset

↳ Import data over pretrained model

→ Data Preprocessing

↳ tokenization

↳ eg I, love, reading

↳ Mapping is done

Data		Numeric
token1	→	vector1
token2	→	vector2
token3	→	vector3

DATE : __/__/__
PAGE :

Data modelling / Training

↳ RNN
↳ content of our data
↳ LSTM
↳ Transformers (GPT)
↓
Generative
Pretrained
Transformers

Accuracy check

Loss ↓
$(y - \hat{y})$ ↓
(actual - predicted) ↓
↳ Optimizer → gradient Descent

learning rate → ~~rate at~~ at how much rate you want error to be reduced.

NLP Pipeline

Data Acquisition

(Collection of data)
(Getting access of data)

↓

Data extraction
(text)

↓

Cleanup

↓

Sentence tokenization

↓

Word tokenization

↓

Stemming / Lemmatization
(mapping words to their base words)

↓

Feature engineering
(words to number)
(Vectorization)
eg TFIDF, One hot encoding)

↓

(Model Building) ML classification (Evaluation)

↓

Deployment

DATE :__/__/__
PAGE :

* Vectorizers

① TF-IDF vectorizer

First what is DF

Document Frequency (DF) = No. of times term t appear in all docs

IDF (Inverse DF) = $\log\left(\frac{\text{Total docs}}{\text{No. Of documents in which t is pres}}\right)$

TF (Term Frequency) = $\left(\frac{\text{Total no. of tims t is present in do}}{\text{Total no of words in doc}}\right)$

TF-IDF = TF × IDF

Label encoding

↳ ~~Conve~~ giving a number to every token present in a document

DATE : __/__/__
PAGE : _____

One hot encoding

→ You have entire ~~toca~~ vocabulary (of words of document) so, you assign an ~~idex~~ index for each individual word where ever, word appears you put 1

I am a boy, He is a boy

	I	He	am	a	boy
I	1	0	0	0	0
am	0	0	1	0	0
a	0	0	0	1	0
boy	0	0	0	0	1

Bag of words

* You have vocabulary of words, we will assign whole corpus of text as index, wherever words of corpus are there, we will increase the count

	I	He	am	a	boy
I am a boy	1	0	1	1	1
He is a boy	0	1	0	1	1

Word embeddings

↳ Similar words have similar vector
Eg word2vec, Glove, fast text
Bert GPT

Vectorization using Deep learning

WORD 2 VEC
- CBOW
- SKIP GRAMS (Pre trained models)

WORD2VEC

→ Algorithm uses a neural network to learn word associations from a large corpus of text.

Steps

↳ Feature representation
  ↳ Every word is represented as a feature vector (according to features)

eg Corpus :-

	King	man	woman
authority	1	0.2	0.2
gender	-1	-1	1
Rich	1	0.5	0.5
Poor	0	0	0

King - man + Woman → Queen

DATE : __/__/____
PAGE : _____

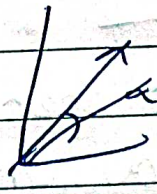
WORD2VEC → helps in understanding semantics of a word

Pre-Trained model

↳ already trained models that use high resources, space and processing.

Cosine similarity

↳ The similarity b/w 2 vectors is determined by the angle made b/w these vectors.



Distance = $1 - \cos\theta$

DATE : __/__/__
PAGE :

LLM (Large language Models)

Base LLM

→ Predicts next word, based on training data

Instruction Tuned LLM

→ Tries to follow instructions

Principle of prompting

(1) → Write clear and specific instruction
↳ Use delimiters " " , ' ' , < > , : .
↳ Check whether condition are satisfied
↳ Give examples

(2) → Give model time to think
↳ specify steps required to complete task
↳ Instruct the model to work on its own solution before going to conclusion