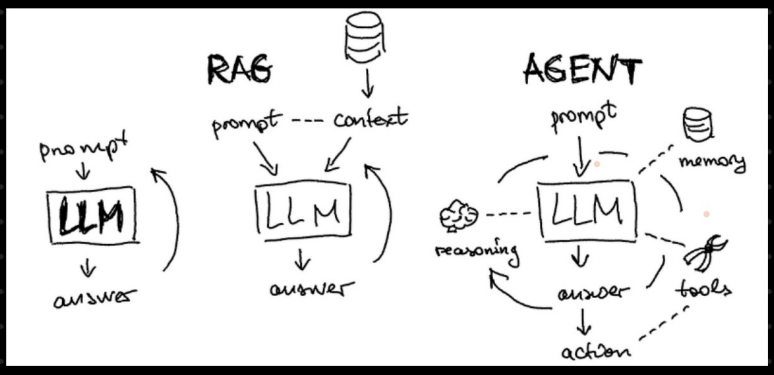
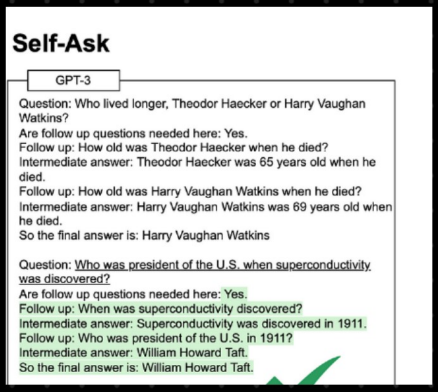
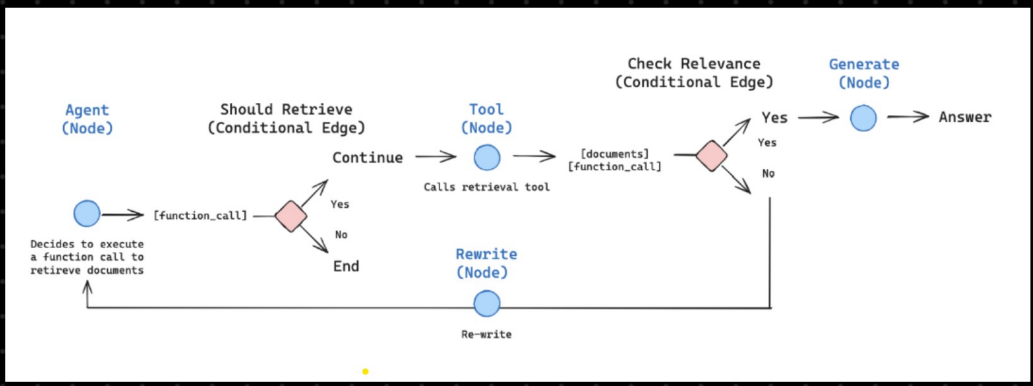
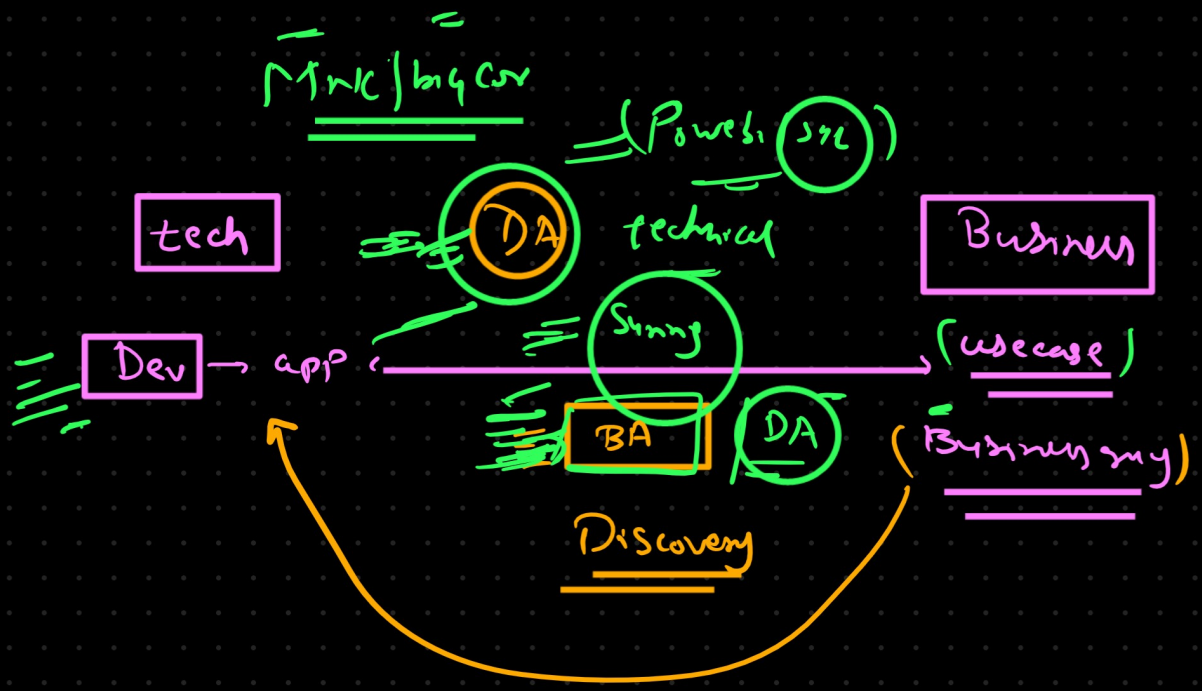

RAG
prompt
LLM
answer
prompt --- context
LLM
answer
AGENT
prompt
memory
reasoning
LLM
answer
tools
action

TAO


Self-Ask
GPT-3
Question: Who lived longer, Theodor Haecker or Harry Vaughan Watkins?
Are follow up questions needed here: Yes.
Follow up: How old was Theodor Haecker when he died?
Intermediate answer: Theodor Haecker was 65 years old when he died.
Follow up: How old was Harry Vaughan Watkins when he died?
Intermediate answer: Harry Vaughan Watkins was 69 years old when he died.
So the final answer is: Harry Vaughan Watkins
Question: Who was president of the U.S. when superconductivity was discovered?
Are follow up questions needed here: Yes.
Follow up: When was superconductivity discovered?
Intermediate answer: Superconductivity was discovered in 1911.
Follow up: Who was president of the U.S. in 1911?
Intermediate answer: William Howard Taft.
So the final answer is: William Howard Taft.


Agent
(Node)
Should Retrieve
(Conditional Edge)
Tool
(Node)
Check Relevance
(Conditional Edge)
Generate
(Node)
Continue
Calls retrieval tool
[documents]
[function_call]
Yes
Answer
Yes
No
[function_call]
Yes
No
Decides to execute a function call to retireve documents
End
Rewrite
(Node)
Re-write


tech
DA
technical
Business
Dev
app
Sunny
usecase
BA
DA
Discovery

(Text Processing)

Analysis

{Punc, stopword, emoji}

Python script

Stopwords — making ↓ sence

~~I~~ ~~am~~ (Sunny Saurabh)

~~who~~ (teach genAI)

[I, am, who]

(Sentences)

NLP ⇒ Text

entire data

1. Paragraph → CORPUS
2. Sentence → Document ⇒ RAG (Doc)
3. words → tokens ⇒ token limit
4. character → character

Chunking

{gpt 3.5, gpt 4, gemini}

Vocabulary → Collection of unique words (tokens)

token ≠ word

Word, Sentence

↳ langchain, Llamaindex.

desired result

Whatever technique

Regex → optional

Custom

Split

NLTK or other libraray

word_tokenize, sent_tokenize

Stemming => root form → not accurate

Lemmetization => original word

Base Form (root)

↓

accurate

Stemming
Definition: Stemming is the process of reducing a word to its root form by removing suffixes or prefixes, often without considering the word's meaning.

Key Characteristics:

Usually rule-based or algorithmic.
The output may not always be a valid word.
Focuses on quick and computationally efficient reduction of words.

Example:

Words like running, runner, and ran might be reduced to the stem run.
Similarly, better might be stemmed to bett.
Common Stemming Algorithms:

Porter Stemmer
Lancaster Stemmer
Snowball Stemmer

Advantages:
Simple and fast.
Useful in scenarios where slight inaccuracies are acceptable, like search engines.
Disadvantages:

Often produces stems that aren't actual words (e.g., studies → studi).

Text Preprocessing

Huge data ⇒ messy, ambiguous data

↑

Your understanding

↑

text Preprocessing

Definition: Lemmatization reduces a word to its base form (lemma) while considering the context and the word's grammatical meaning (e.g., part of speech).

Key Characteristics:

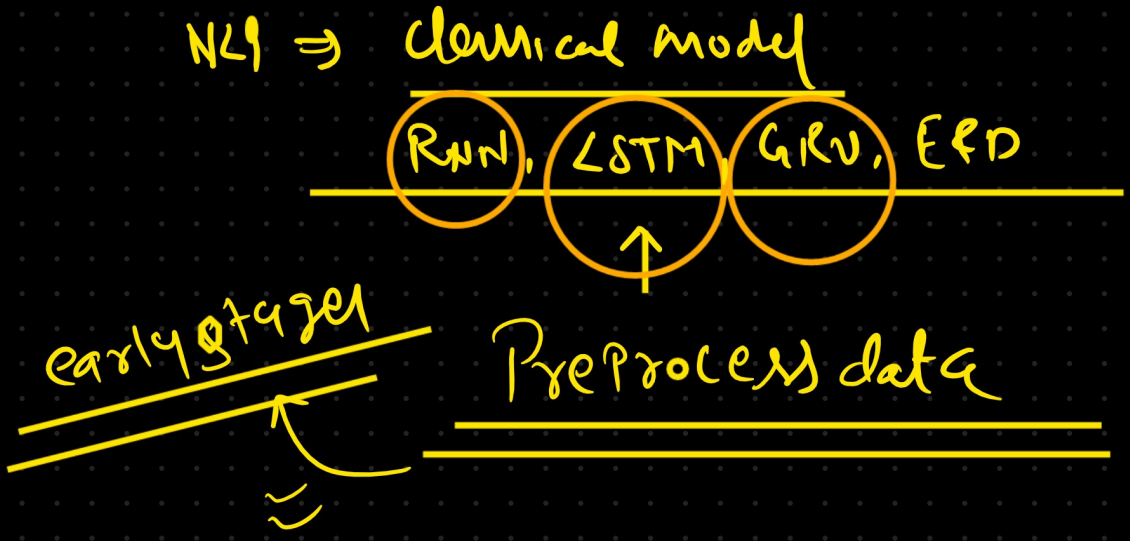
More sophisticated and linguistically informed.
The output is always a valid word.
Requires tools like a dictionary or vocabulary resource to find the base form.
Example:

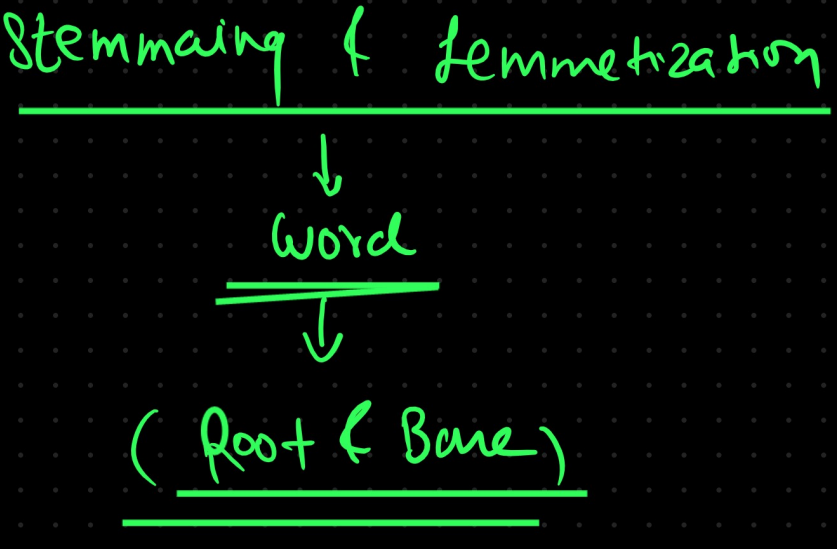
running → run
better → good (context-aware, since "better" is the comparative form of "good").
Common Lemmatization Tools:

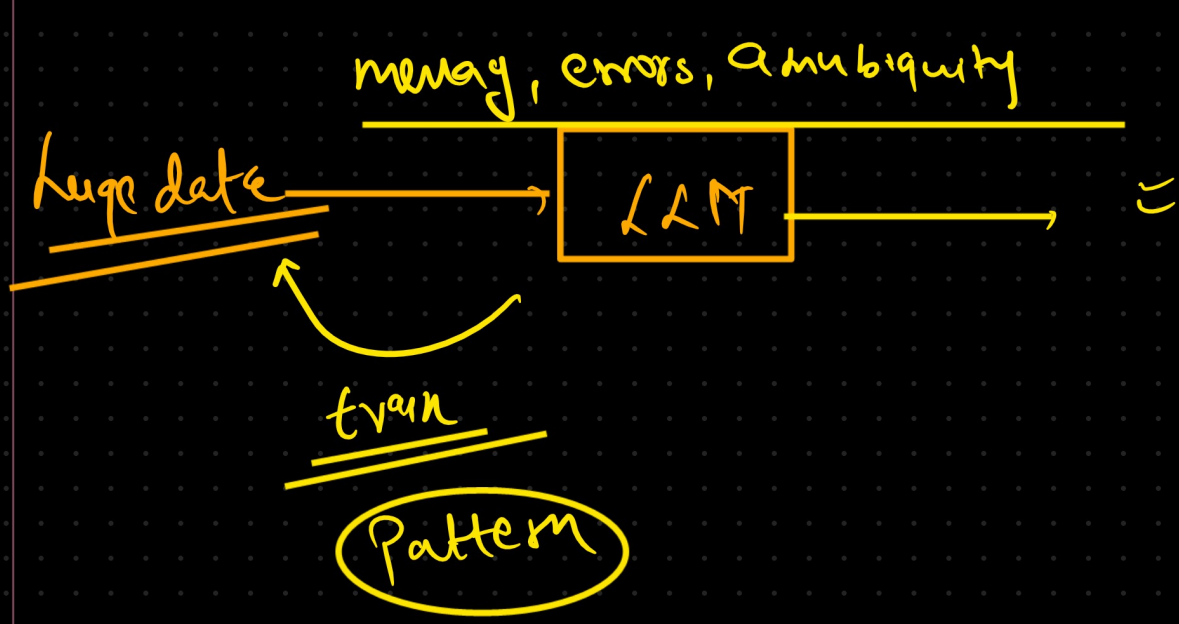
WordNet Lemmatizer (in NLTK)
spaCy (built-in lemmatizer)
Advantages:

Produces more accurate base forms.
Useful in linguistically rigorous tasks like machine translation or sentiment analysis.
Disadvantages:

Slower and computationally more expensive than stemming.
Requires additional information, such as part-of-speech tags.







Q. IS text Preprocessing req. for LLM?

Most of time → NO

but if data having lot's of error in that case YES

{ Punc, Stopword, ambiguous & redundant word, emoji }

tokenizer ⇒ Langchain

Encoding & Embedding

= vector

{ Numeric representation of text }

Text → Number

Encoding => frequency based method

Embedding => Neural network based method

=> text → Number → vector

= Set of number

Encoding (classical method) — OLD

lookup to SMP

1. OHE
2. BOW
3. TF-IDF
4. N-GRAMS
5. Custom method

Embedding (classical) = OLD

1. Word2vec

Word to vector → Neural network

Glove → Math (matrix factorization)

State of Art technique (New)

transformer Arch

Sentence embedding

1. BERT
2. ELMo
3. Gemini
4. openAI
5. MPT

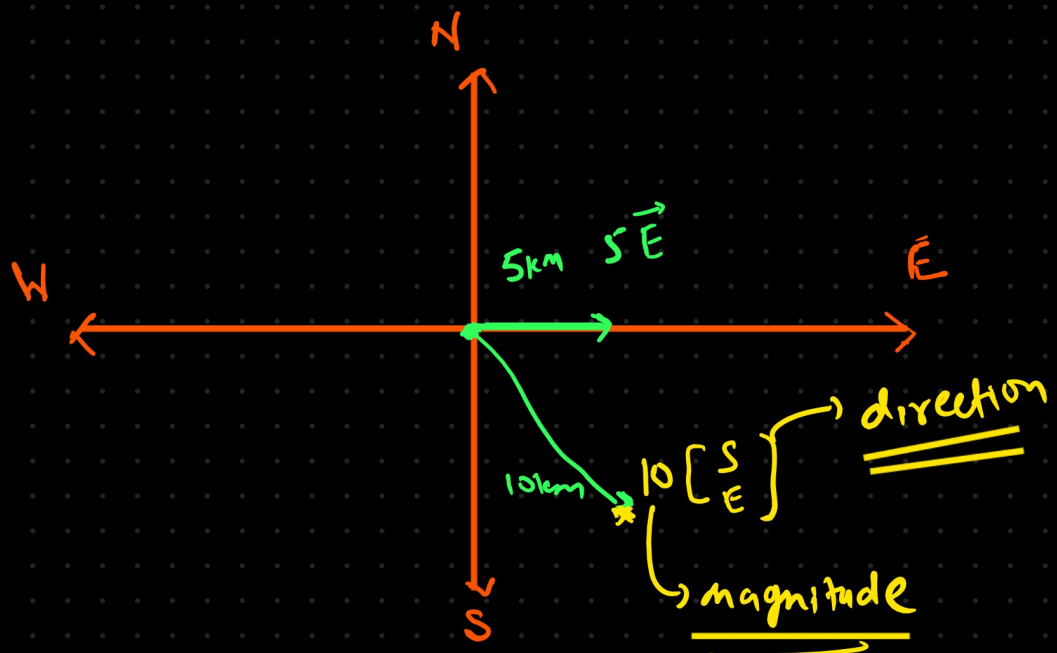
(Senfence transformer) ⇒ huggging

Sentence ⟶ embedding (Numric)

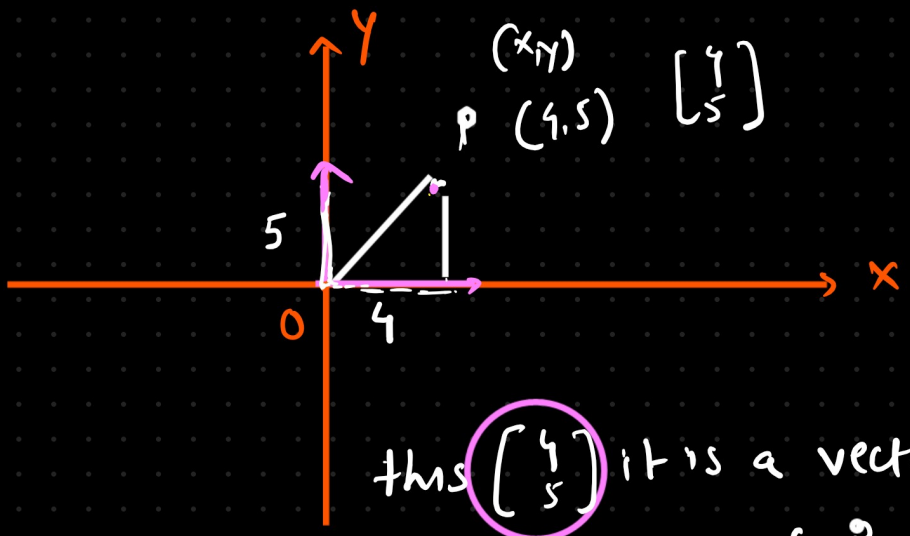
vector (set of Num)

Data → ST → Transformer → Sentence embedding

Vectors



Physics ⇒ M + D = V



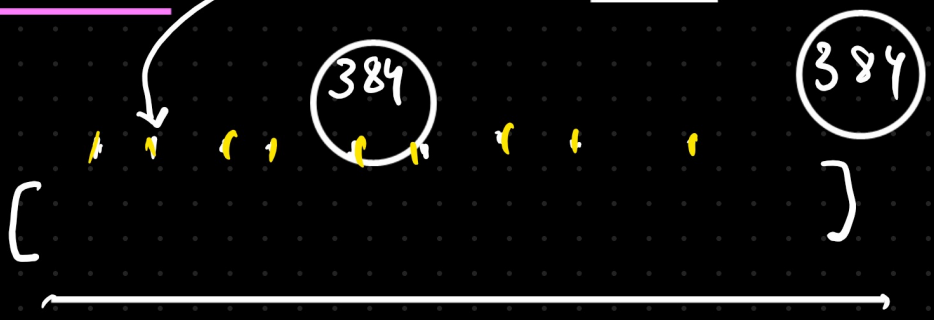
this $\begin{bmatrix} 4 \\ 5 \end{bmatrix}$ it is a vector representation of Point P

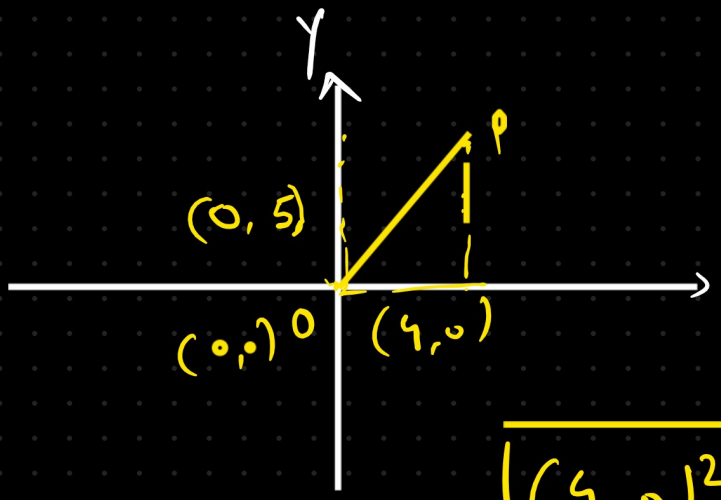
? → (P → [4,5])

2D

2dim → x, y

Word 2 vec ⇒ Sunny ⇒ 384 ↗ by default





$\overrightarrow{OP}$ = M + D

=

$\sqrt{(4-0)^2 + (5-0)^2}$

$\sqrt{16+25}$

$\sqrt{41}$

$\overrightarrow{OP} = \sqrt{41}\begin{bmatrix} x \\ y \end{bmatrix}$

$\overrightarrow{OP} = \sqrt{41}\begin{bmatrix} 4 \\ 5 \end{bmatrix}$

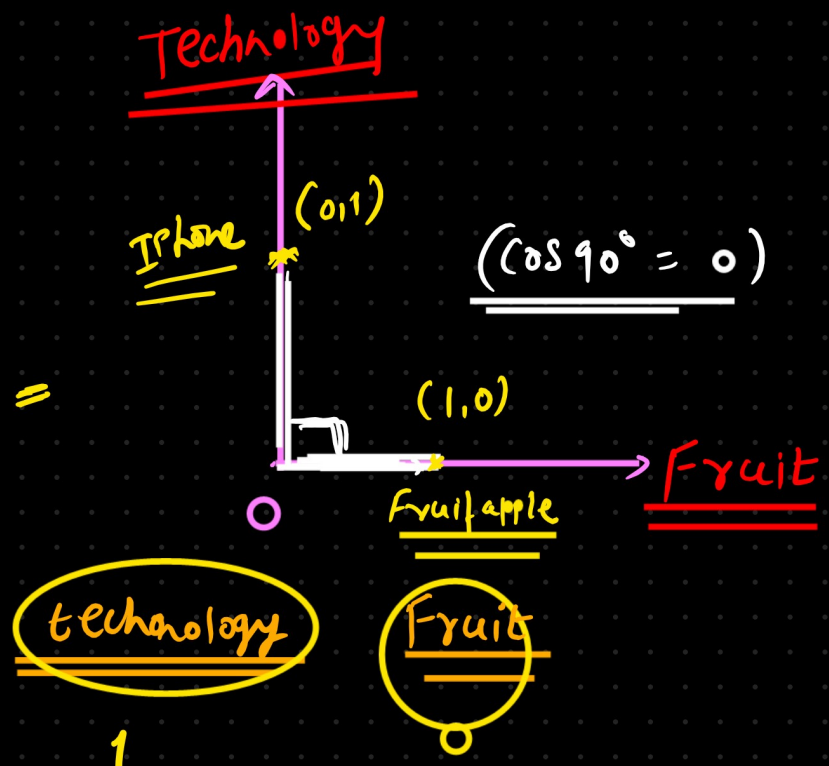
$OP = \sqrt{41}$ = $OP = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$

Dimension

OP or P $\begin{bmatrix} 4, & 5 \\ x & y \end{bmatrix}$

Apple

1. Iphone
2. Fruit Apple =



Iphone

1

Iphone → [1,0] ← vector

Apple → [0,1] ← vector

Similimty Search

1. Dot Product
2. Cosine Simi.

[1,0]
[0,1]

$(x_1 y_1)\ (x_2 y_2)$

$(x_1 * x_2) + (y_1 * y_2)$

$(1 * 0) + (0 * 1)$

$0 + 0 = 0$

RAG ⇒ Semantic Search

1. OHE
2. BOW
3. TF-IDF
4. N-GRAMS
5. Word2Vec

Sat

Practical

Sun

RNN vs LSTM vs EED with attention vs transformer

= Sat.

= transformer

1. OHE

Ex:-

D_1	People	watch	Cricket-
D_2	Cricket	watch	Cricket-
D_3	people	give	Comment-
D_4	Cricket-	give	Comment-

Document-

OHE

Vocabluary → { People, watch, cricket, give, comment }

V = 5

People	watch	cricket	give	comment

D_1 [[1 0 0 0 0]
[0 1 0 0 0]
[0 0 1 0 0]]

3 x 5

D_2 → [[0 0 1 0 0]
[0 1 0 0 0]
[0 0 1 0 0]]

5 x 5

Dimension → 3 x 5

Pros

1. Easy to implement

Cons

1. Sparse matrix
   → [too many zeros in the given matrix]
2. OOV
3. No fixed dimension

Vector ⇒ set of Numbers

→ [1, 2, 3, 4, 5]

BOW ⇒ 2018, 2019 → Frequency based method (occurance)

→ D_1	People	watch	Cricket	→ vocab ⇒ Distinct words.
D_2	Cricket = People	watch	Cricket = Cricket	⇒ {People, watch, Cricket, give, Comment}
D_3	People	give	Comment	
D_4	Cricket	give	Comment	

	People	watch	Cricket	give	Comment
D_1	1	1	1	0	0
D_2	0	1	2	0	0
D_3	1	0	0	1	1
D_4	0	0	1	1	1

→ $D_1 \rightarrow [1,1,1,0,0]$

Advantage

1. Simple

Dis-adv

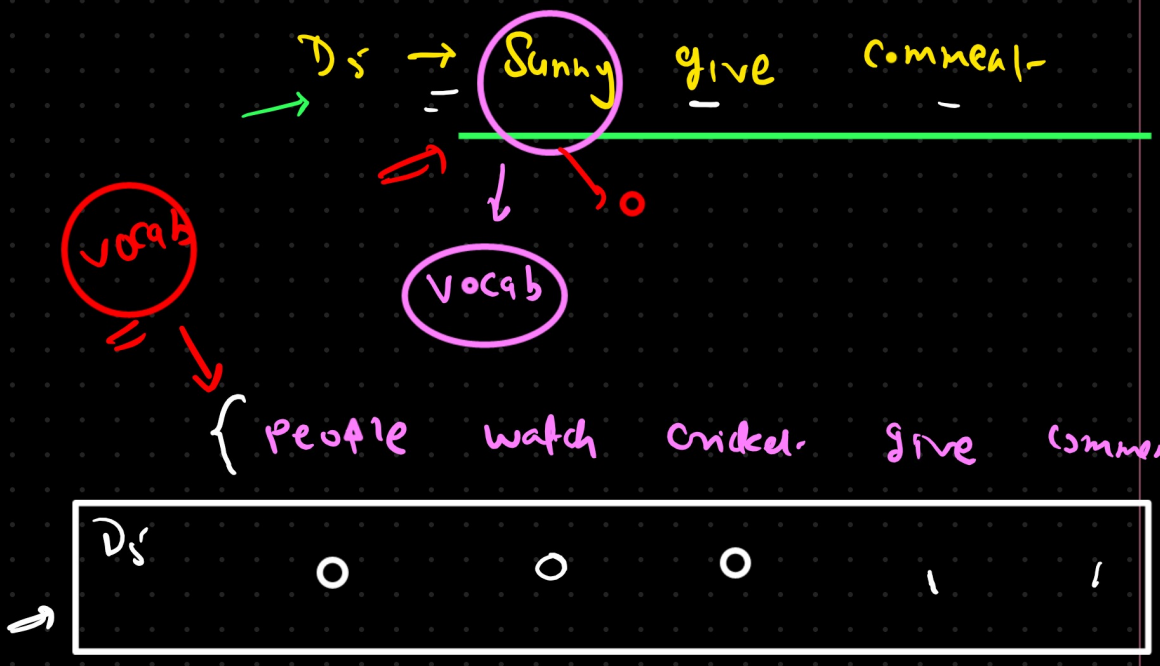
1. sparsity

Zeros, ones.

capture → 2. not Capturing (Semantic info)

3. (OOV)

out of vocab



OOV → if any new (sent / Doc)

(new words)

(Existing voccab)

(loosing info)

(Update vocab)

(BOW, OHE)

N-grams 2018-2019

BOW

- N=1 ⇒ Single word (uni gram)
- N=2 ⇒ Pair of 2 words (Bi-gram)
- N=3 ⇒ Pair of 3 words (tri-gram)
- ⋮

Pair of the words ⇒ Content of the data

retain the info.

semantic meaning

D_1	People	watch	Cricket
D_2	Cricket	watch	cricket
D_3	People	give	comment
D_4	cricket	give	Comment

~~N=1,2,3~~

N=3

{people, watch, cricket, give, comment}

Single words

Bi gram → double words

Vocab

{(People, watch), (watch, cricket), (cricket, give), (give, comment)}

(People, cricket) (People, give) (People, comment)

(watch, give) (watch, comment) (Cricket comment) }

↓

→ Semantic

	(P,W)	(W,C)	(C,G)	(G,Co)	(P,C)	(P,G)	(P,Co)	(W,G)	(W,Co)	(C,Co)
D_1	1	1	0	0	1	0	0	0	0	0
D_2	0	1	0	0	0	0	0	0	0	0
D_3	-	-	-	-	-	-	-	-	-	-
D_4										

Advantage

1. Simple to use
2. Sustain context (pair)

Disadvantage

⇒ 1. Sparse matrix (0,1)

2. OOV issue

↓

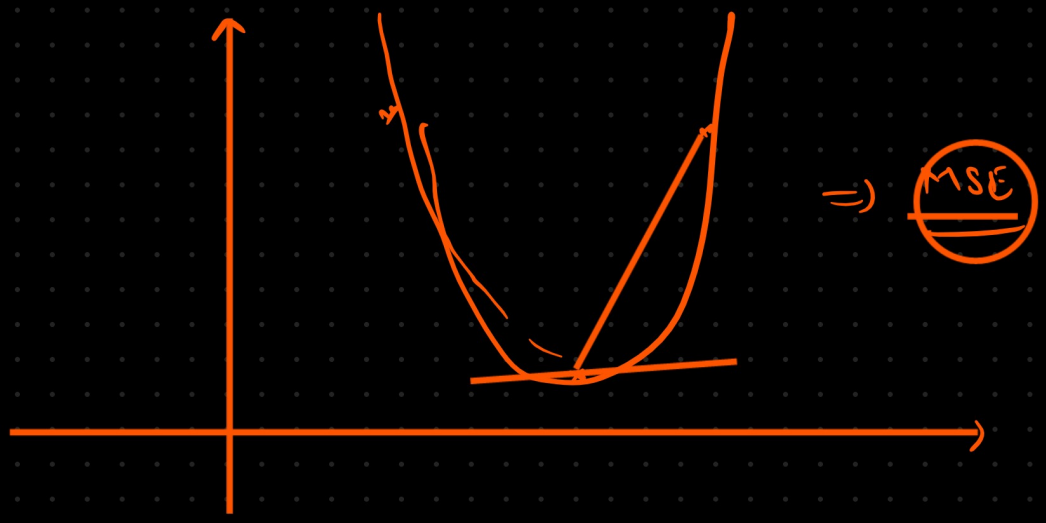
new word (old vocab)

Upcoming Sat-

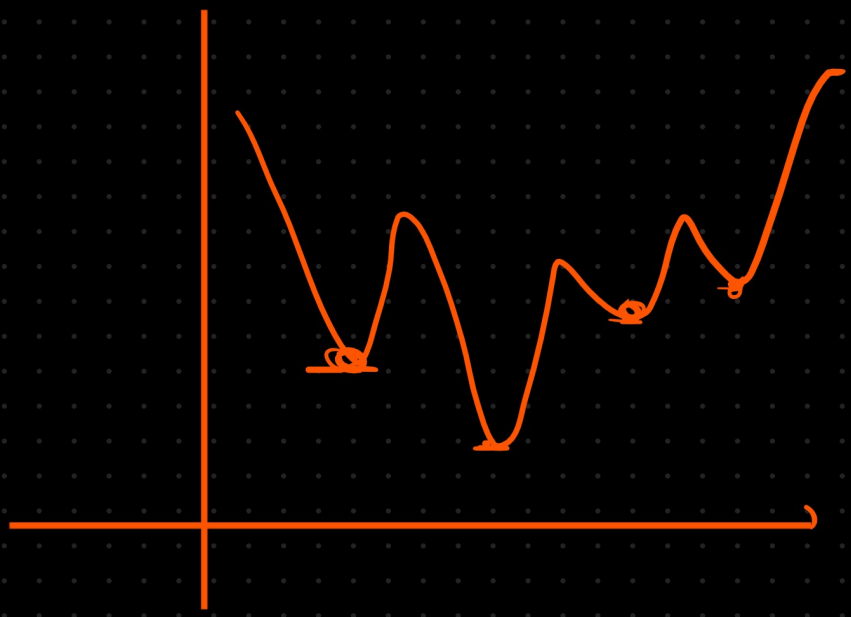
1. TF-IDF
2. Word2vec

Practical of both

huggingface sentence embedding



non-convex loss fn



1. Optimizers with momentum
2. Learning
3. Batch normalization
4. Early stopping
5. (Reg + Loss)
6. Gradient clipping

1. TF-IDF → BM25 → RAG (Reranking)

Theory & Practical

2. word2vec

word → vec

theory & Practical

3. Sentence embedding → (huggingface)

↓

Practical

Sentence → vec

(Avg word2vec)

Search engine → google → 2009/2010

TF-IDF → word

(Term frequency)

(Inverse document-freq.) → Sentence

Data → vector

↳ numerical representation of text

(corpus) → Sentence → (words) → NLU

Sent token (D_1, D_2, D_3)

Data

	(1/3)	(1/3)	(1/3)
D_1	People	watch	cricket
D_2	cricket (2/3)	watch (1/3)	cricket (2/3)
D_3	People (1/3)	Give (1/3)	Comment (1/3)
D_4	cricket	Give	Comment
	(1/3)	(1/3)	(1/3)

Document == Sentence

$$\text{Term freq}(\text{word}, D) = \frac{\text{occurrance of word in given sent (Doc)}}{\text{total word in given sent.(Doc)}}$$

$$\text{IDF} = \log\left(\frac{\text{No. of total Sentencey (Doc)}}{\text{No. of Sentence with the given word}}\right)$$

		$\log(4/2)$	$\log(4/3)$	$\log(4/3)$
D_1	→	People	watch	Cricket
D_2		Cricket	watch	cricket
D_3	→	People	give	Comment
D_4		cricket	give	Comment

		$\log(4/2)$	$\log(4/2)$

Data → Encoding & vector & number

TF-IDF

[TF X IDF]

Vocab ⇒ Unique words from the data

	Cricket	Comment	give	People	watch
D_1	$\frac{1}{3} \times \log(4/3)$	0	0	$\frac{1}{3} * \log(4/2)$	$\frac{1}{3} * \log(4/2)$
D_2	$\frac{2}{3} * \log(4/3)$	0	0	0	$\frac{1}{3} * \log(4/2)$
D_3	0	$\frac{1}{3} \times \log(4/2)$	$\frac{1}{3} * \log(4/2)$	$\frac{1}{3} * \log(4/2)$	0
D_4	$\frac{1}{3} * \log(4/3)$	$\frac{1}{3} * \log(4/2)$	$\frac{1}{3} * \log(4/2)$	0	0

['comment' 'cricket' 'give' 'people' 'watch']

Key takeaway

1. TF → Importance of word in given sent-
2. IDF → Uniqueness of word across all the sentences
3. log → to compress or normalize the IDF value or reduce the skewness in IDF value.
4. (TF)*(IDF) → we are reducing the weightage of more common word and giving importance to unique (rare) words

(TF) (IDF)

(uniqueness)

Advantage

1. Simple
2. Content rich
3. able to identify uniqueness & importance of word

Disadvantage

OOV (huge data)

→ Sparsity

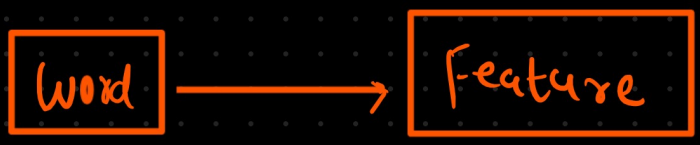
Encoding ⇒ frequency (word) ⇒ Sparse matrix [0,1]

Embedding ⇒ frequency X ⇒ Dense matrix (very less 0,1)

Neural network

Word2 vec

word ⟶ vec



= Google word2vec model ⇒ Neural Network

Google news data ⇒ 2012-13

(300-dimension)

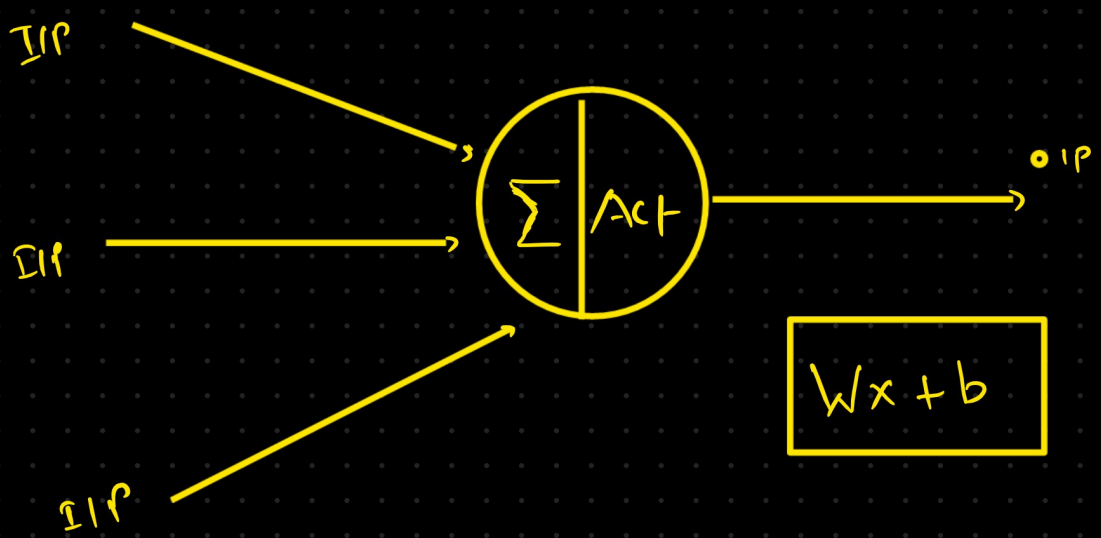
(retraining is possible)

OOV

Customdata

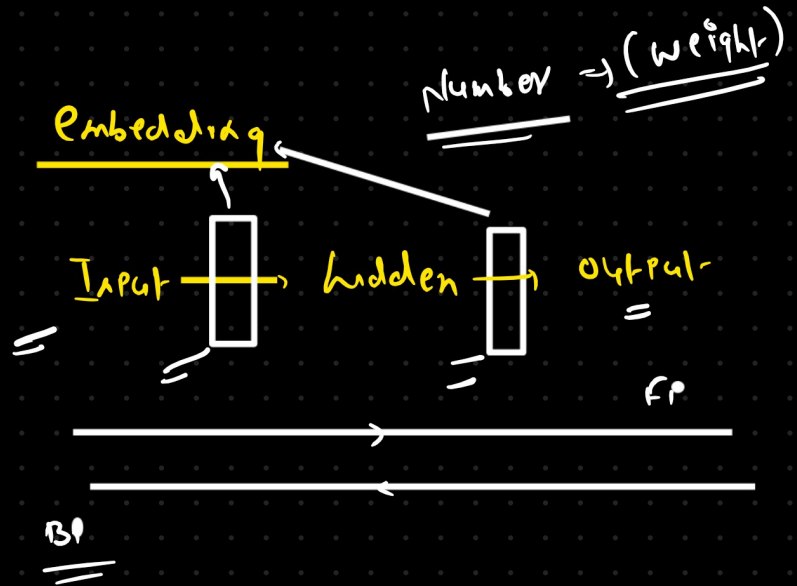
Neural network

1 Perceptron



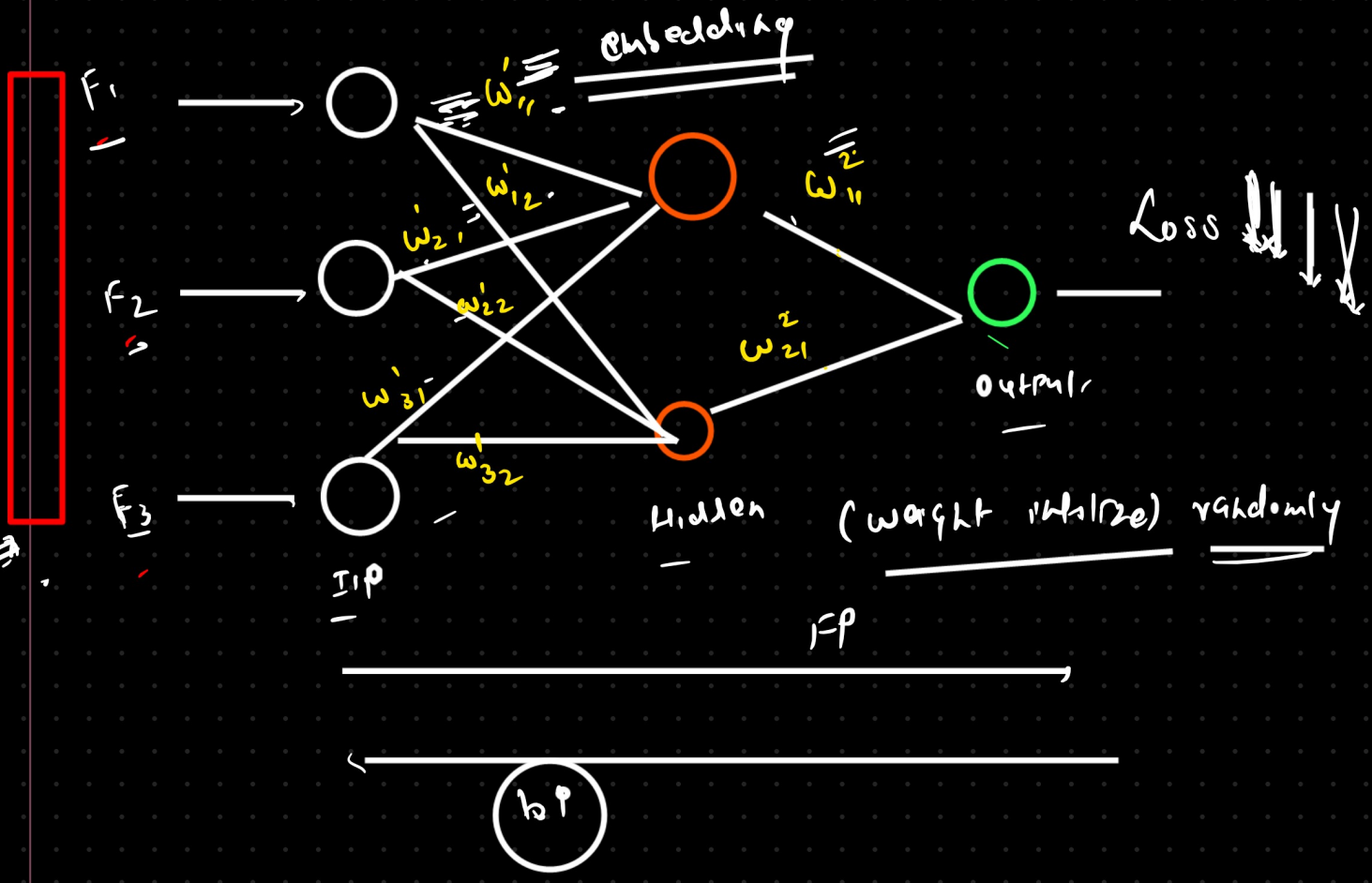
2 Multilayer Perceptron

1. Input
2. Hidden
3. Output



House

-	F1 Size	F2 location	F3 price	decision Y/N
[-	-	-]		



Feature [ ]

text → feature

Features

Context window

Neural network => weight ← output

word ⟶ feature

[0 - 1]

	KING	SPEAK	RUN	RICH	TEACHER
Sunny	0	1	0.8	0.2	1
TIGER	0	0	0.95	0	0

word

manully → value

NN => weight

Google

feature => BOW (X)

NN weights

2 technique

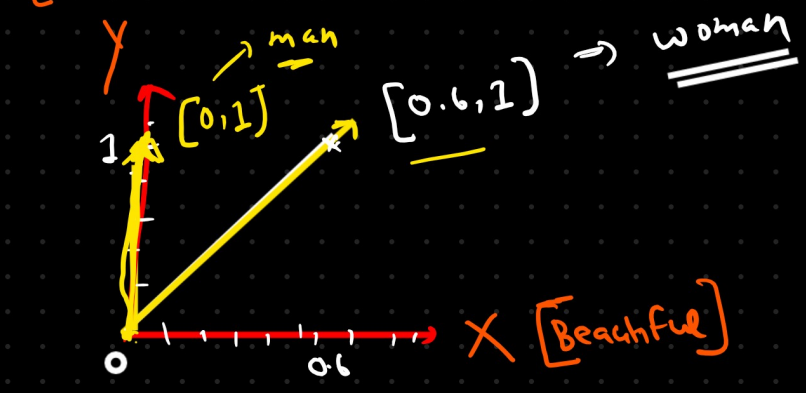
RNN, LSTM, transformer

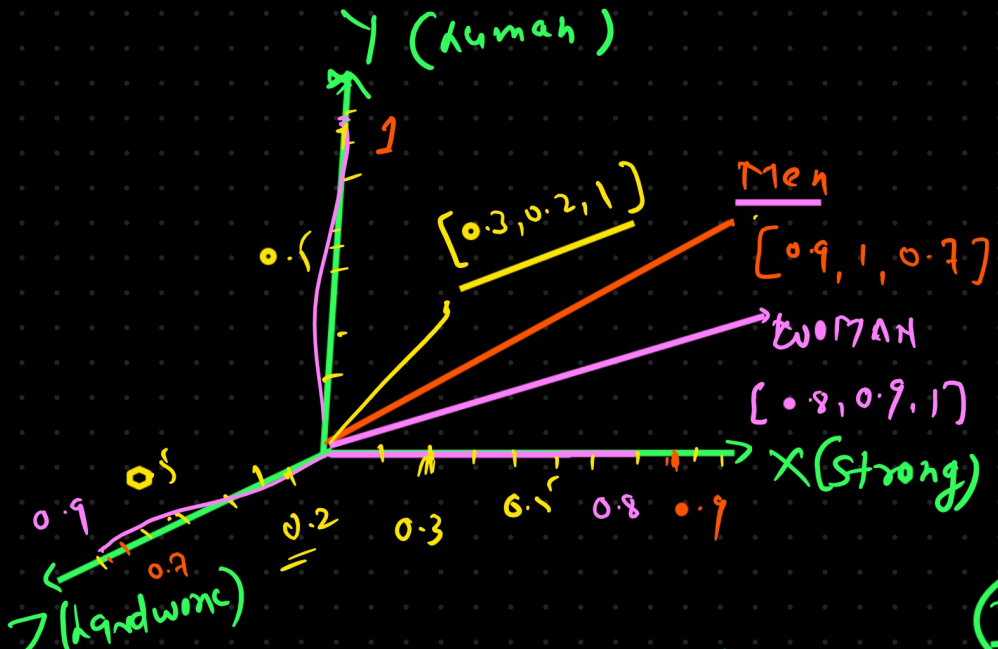
Vector ⇒ 2D [x,y]

3D [x,y,z]

4D [x,y,z...]

3000-D [x,y,z .....300]





Men
woman
child

Transformer

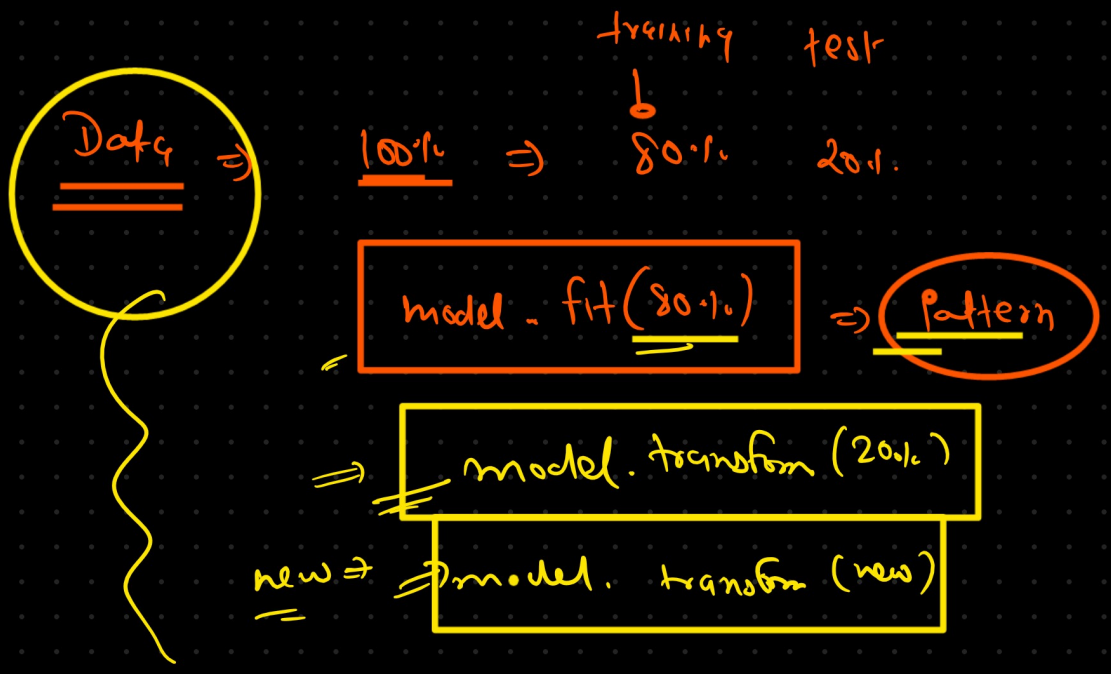
⇒ (Skipgram, cbow)

⇒ 2 method

Custom training

Running custom vs transformer

Running



Preprocessing

train test

800 records

200 records