

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.

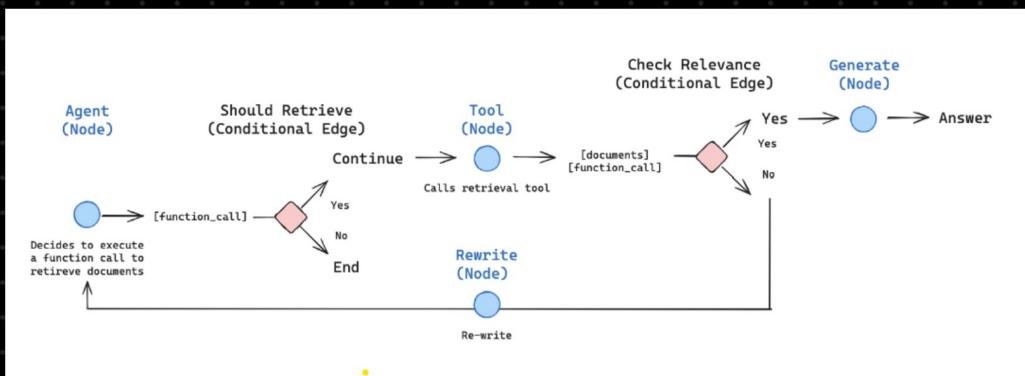
Follow up: When was superconductivity discovered?

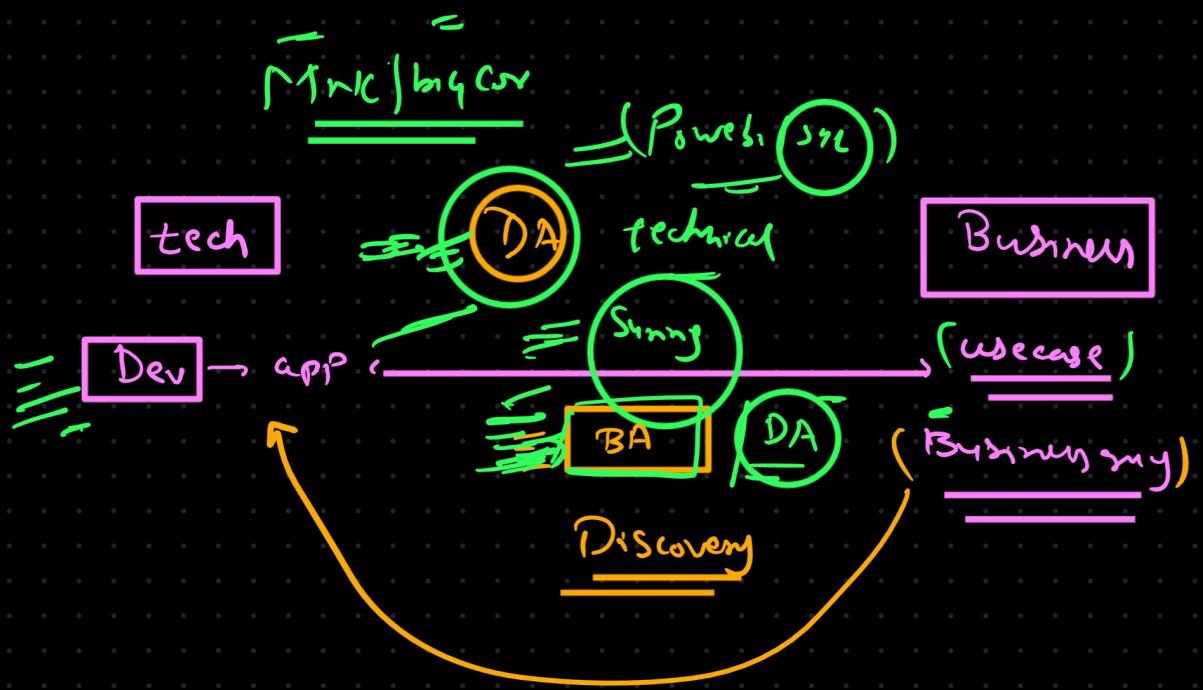
Intermediate answer: Superconductivity was discovered in 1911.

Follow up: Who was president of the U.S. in 1911?

Intermediate answer: William Howard Taft.

So the final answer is: William Howard Taft.





(Text Processing)

Analysing

= { Punc, Stopword, emoji }

Python script

Stopwords making
sense

- ~~I am (Scary Scavenger)~~
~~who (teach generate)~~

[I, am, who]

(Sentences)

NLP → Text

entire data

= 1 Paragraph → CORPUS

= 2 Sentence → Document

chunking

RAG (Doc)

= 3 Words → tokens → token limit

= 4 Character → character

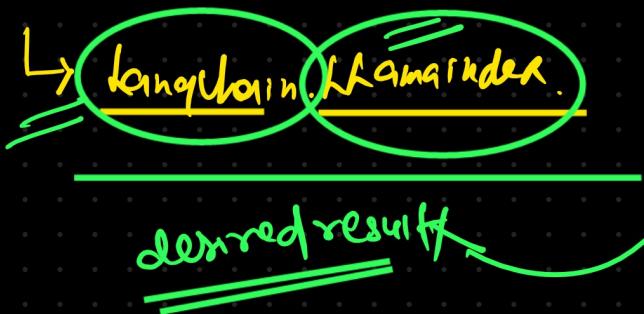
= {
gpt 3.5
gpt 4
gemini }

Vocabulary

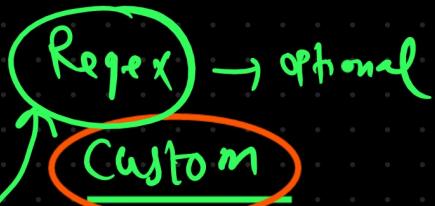
→ Collection of unique words (tokens)

token ≈ word

Word, Sentence



Whatever technique



Split

HTK or other library



Stemming \Rightarrow Root Form $\xrightarrow{\text{not accurate}}$

Lemmatization \Rightarrow original word

Bare Form (not)

\downarrow
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 \Rightarrow 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:

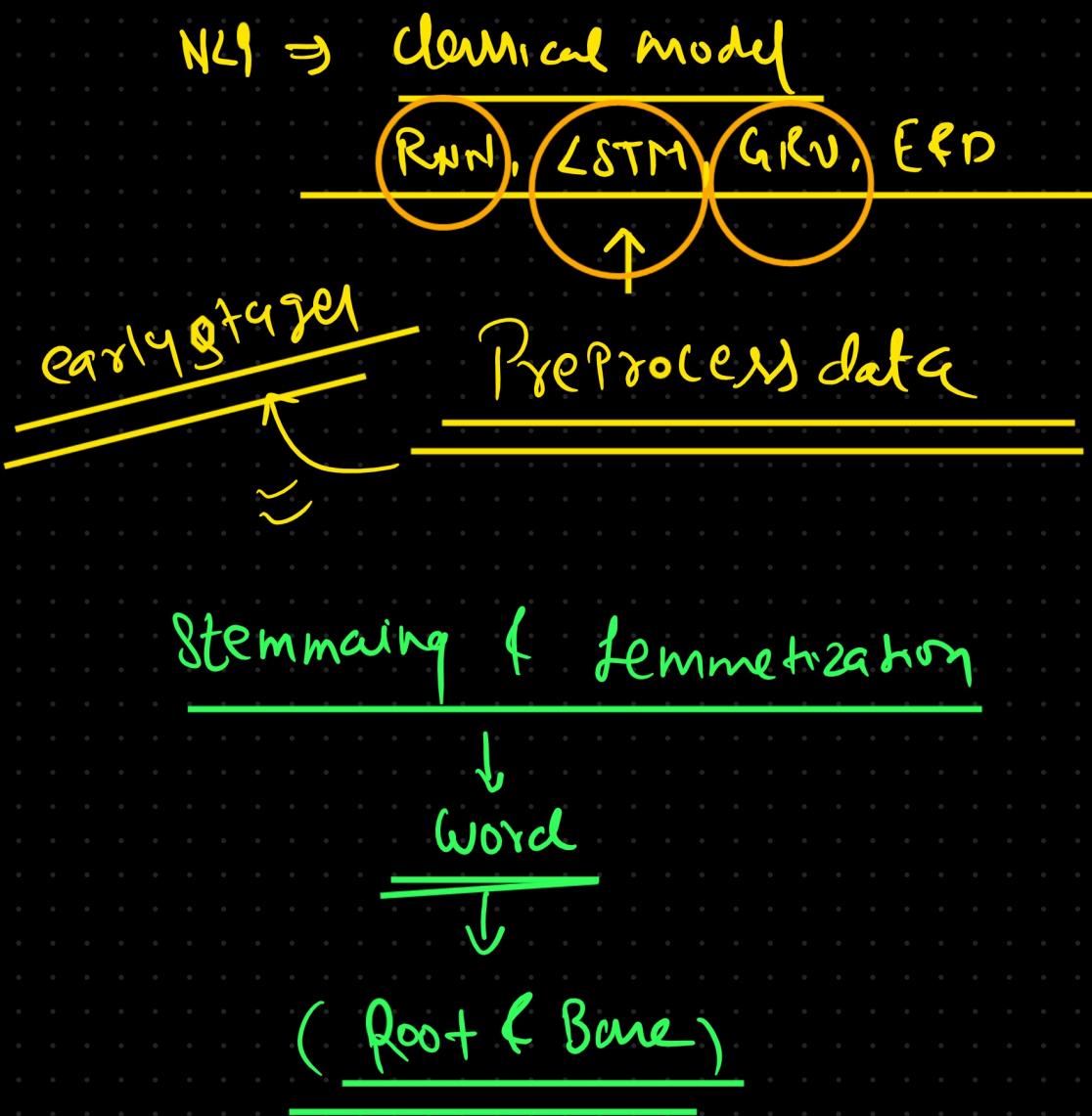
Produces more accurate base forms.

Useful in linguistically rigorous tasks like machine translation or sentiment analysis.

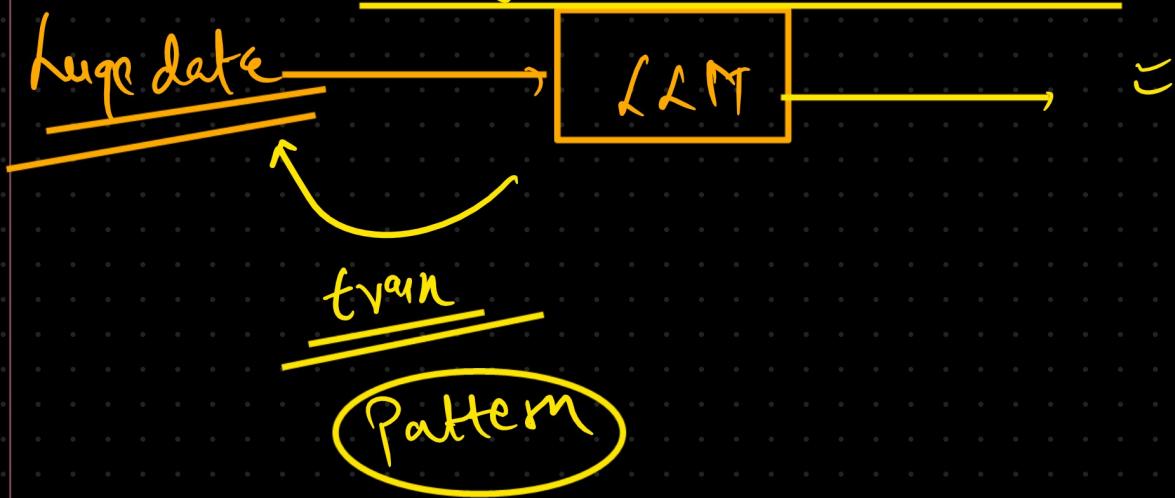
Disadvantages:

Slower and computationally more expensive than stemming.

Requires additional information, such as part-of-speech tags.



memory, errors, Ambiguity

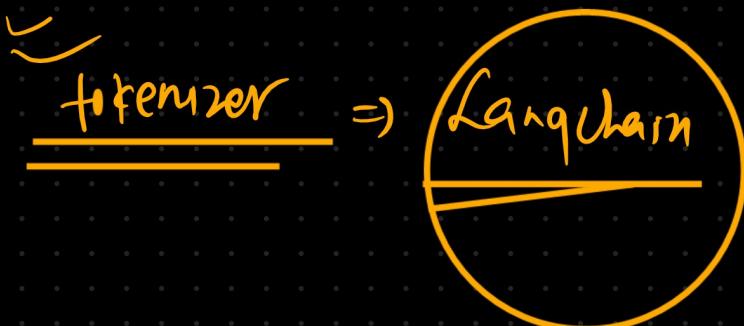


Q. IS text Preprocessing req. for LLM?

Most of time → **No**

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

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



Encoding & Embedding

=

=

vector

{ Numeric representation of text }

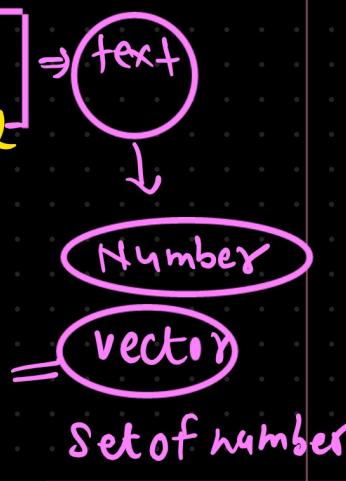


Encoding \Rightarrow frequency based method

embedding \Rightarrow neural network based method

Encoding (classical method)

lookuptable
Skip



① OHE -

② BOW -

③ TF-IDF -

④ N-GRAMS -

⑤ Custom method

Embedding (classical)

① Word2vec OK

Word into vector

↓
neural network

Glove \rightarrow Math

(matrix factorization)

State of Art technique (New)

transformer Arch

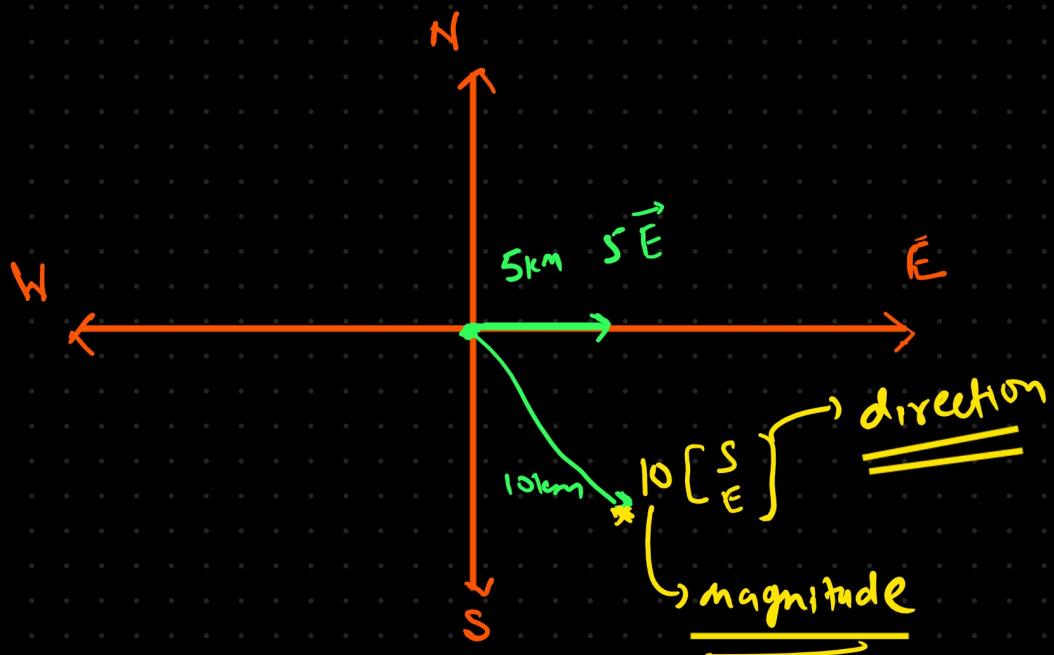
Sentence embedding



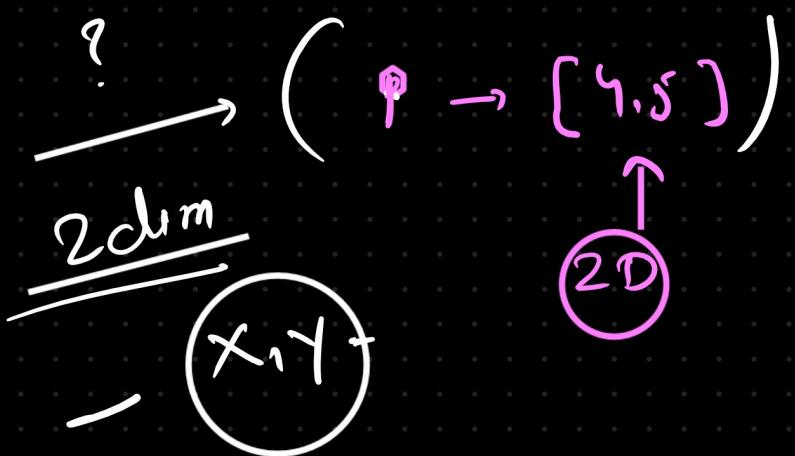
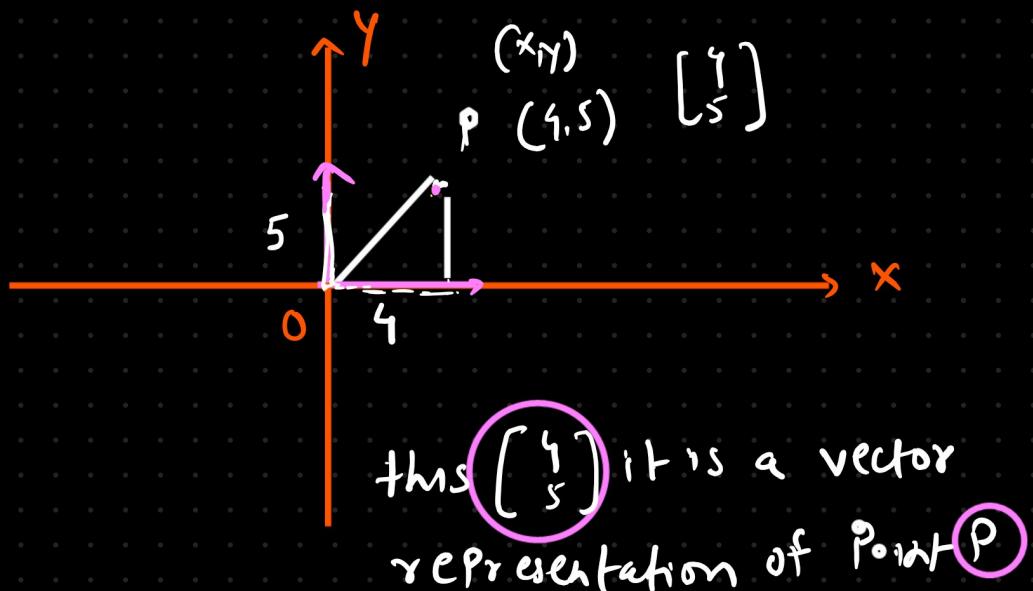
Sentence → embedding (Num^{nic})
vector (set of Num)

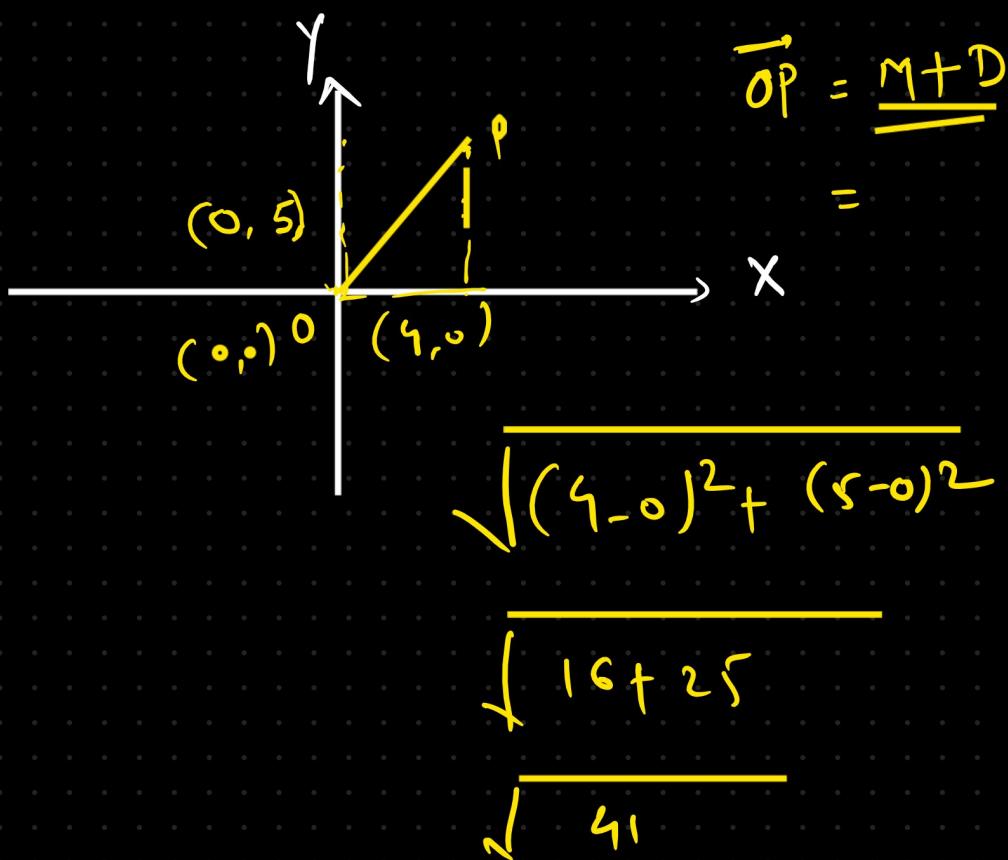
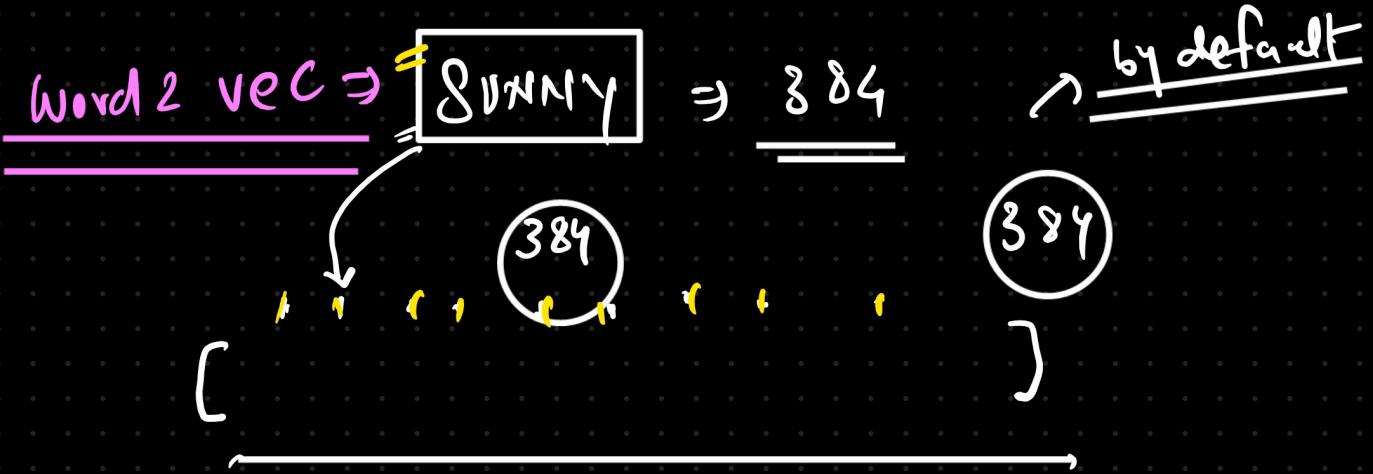
Data → ST → Transformer → embedding
Sentence

Vector &



$$\text{Physics} \Rightarrow M + D = V$$





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

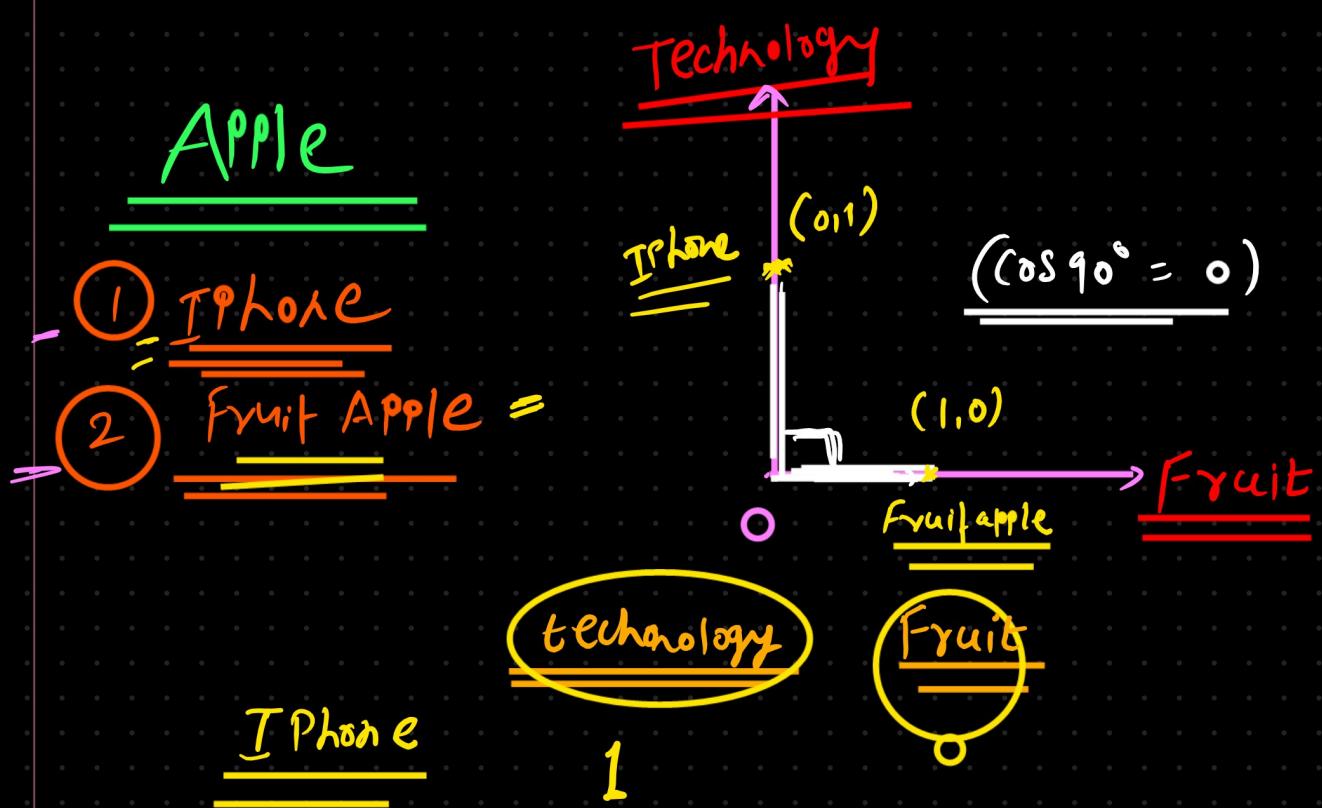
$$\overrightarrow{OP} = \sqrt{41} \begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

$$\text{O } \overrightarrow{OP} = \sqrt{41}$$

$$= \overrightarrow{OP} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

Dimension

$$\overrightarrow{OP} \text{ or } P \left[\begin{array}{c} 4 \\ 5 \\ x \\ y \end{array} \right]$$



iPhone \rightarrow [1,0] \leftarrow vector

Apple \rightarrow [0,1] \leftarrow vector

Similarity Search

- 1
- 2

1 of Product-
Cosine Simi.

$$[1,0]$$

$$[0,1]$$

$$(x_1, y_1) \cdot (x_2, y_2)$$

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

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

0 + 0 = 0

RAG → Semantic Search

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

=

Sat
Practical

Syn

RNN vs LSTM vs
BERT with attention
ViS transformer

- Sat
- transformer

① OHE

Ex:- → D₁. people watch Cricket-
 D₂ Cricket Watch Cricket-
 D₃ people give Comment-
 D₄ Cricket give Comment-
 =

Document-

↓
OHE

Vocabulary → { People, Watch, cricket, give, comment }

V = S

People	watch	cricket	give	comment
--------	-------	---------	------	---------

$$D_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

3×5

$$D_2 \rightarrow \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

5×5

Dimension 3×5

Pros

- ① Easy to implement



Cons

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

Vector \Rightarrow set of numbers
 $\rightarrow [1, 2, 3, 4, 5]$

BOW \Rightarrow 2018, 2019 \rightarrow frequency based method
occurrence

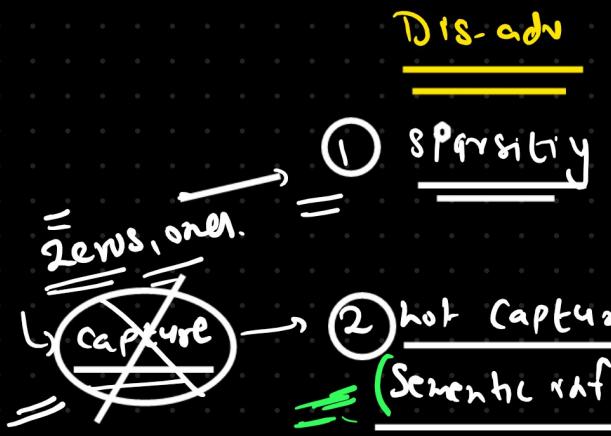
$\rightarrow D_1$ People watch Cricket \rightarrow vocab \Rightarrow Distinct words.
 D_2 Cricket watch Cricket
 D_3 People give Comment
 D_4 Cricket give Comment
 $\Rightarrow \{ \text{people, watch, cricket, give, comment} \}$

	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

$\rightarrow D_1 \rightarrow [1, 1, 2, 0, 0]$

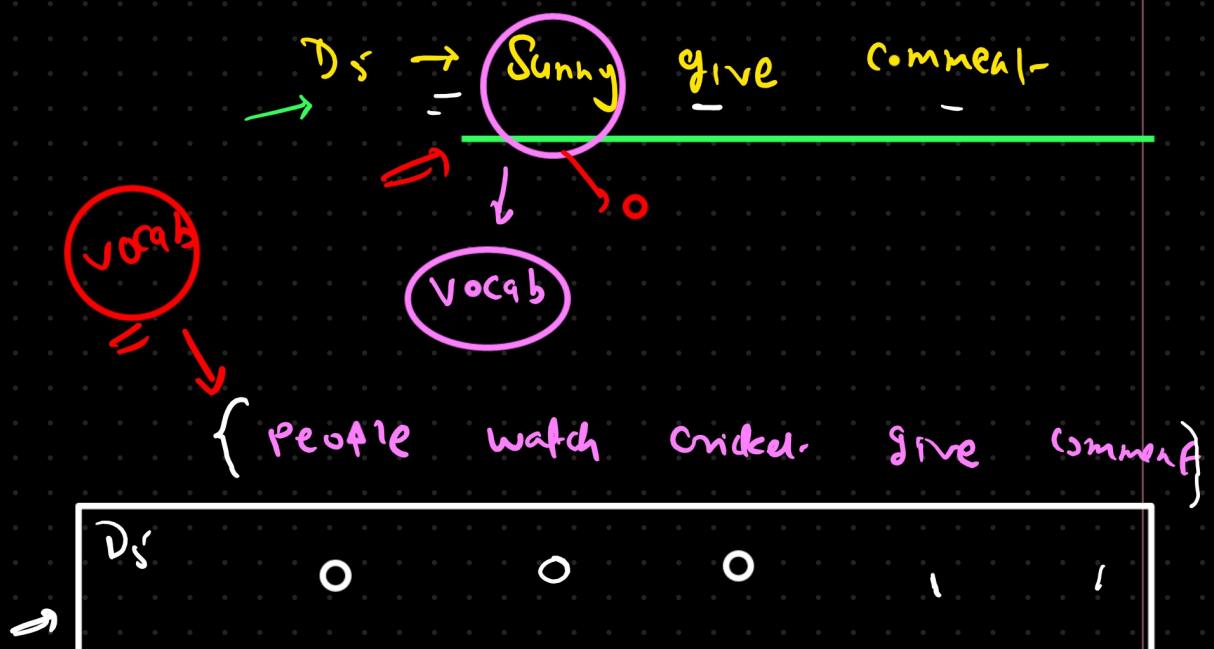
Advantage

1 Simple



③ (OOV)

out of vocab



OOV → if any new (sent / Doc)

(new words)

(Existing vocab)
(losing info)

(update vocab)

(BOW, n-grams)

N-grams

2018 - 2019

BOW

- $N=1 \Rightarrow$ Single word (unigram)
- $N=2 \Rightarrow$ Pair of 2 words (Bi-gram)
- $N=3 \Rightarrow$ Pair of 3 words (tri-gram)

Pair of the words \Rightarrow Content of the data

retain the info.

Semantic meaning

	1	1	
D ₁	People	watch	Cricket-
D ₂	Cricket	watch	comment
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 \rightarrow double words

Vocab
Double words
{ (people, watch), (watch, cricket), (cricket, give), (give, comment) }
(people, cricket) (people, give) (people, comment)

(watch, give) (watch, comment) (click, 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

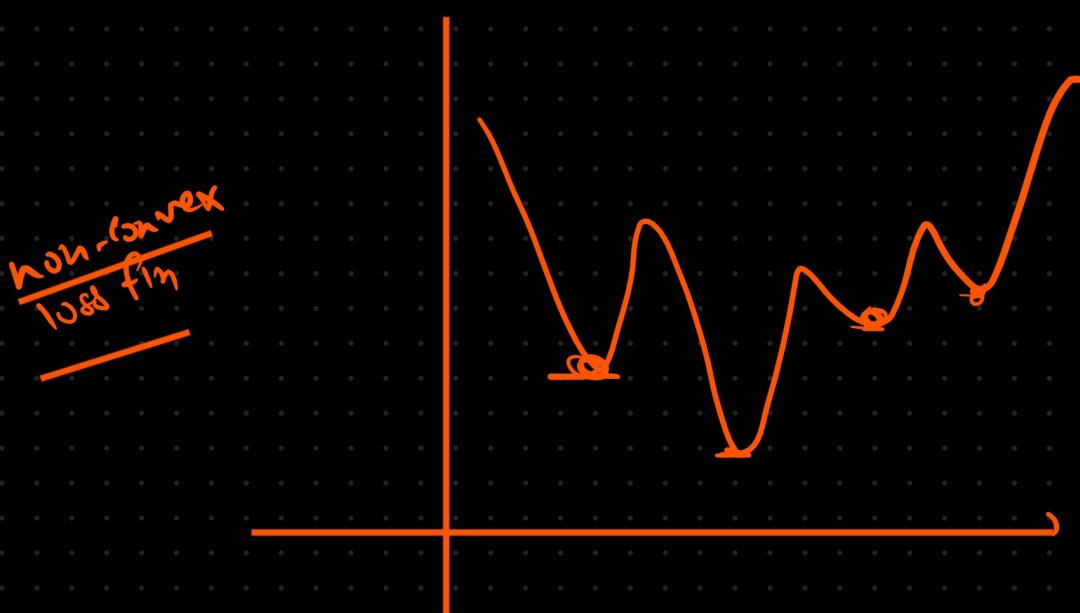
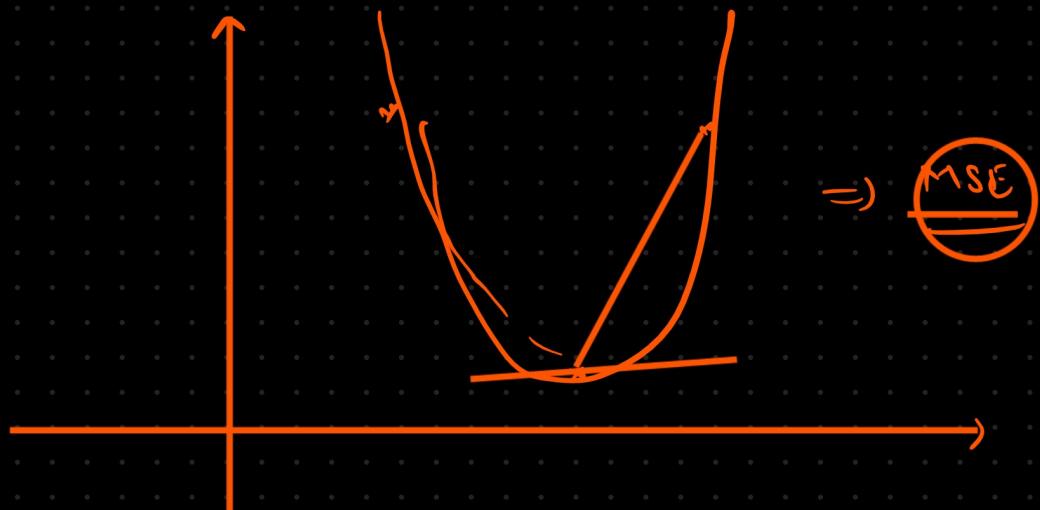
↓
new word (~~old vocab~~)

Upcoming Sali

- ① TF-IDF
- ② word2vec

Practical of both

huggingface sentence embedding



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