Vector Embeddings and RAG

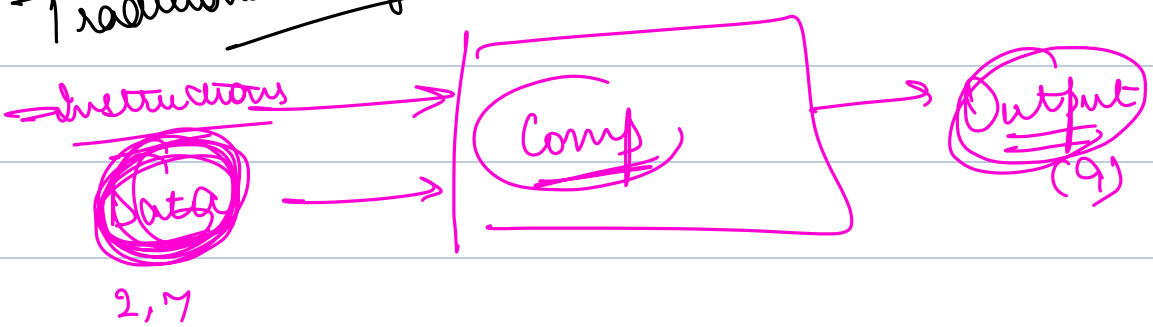
≈ 9:30 PM

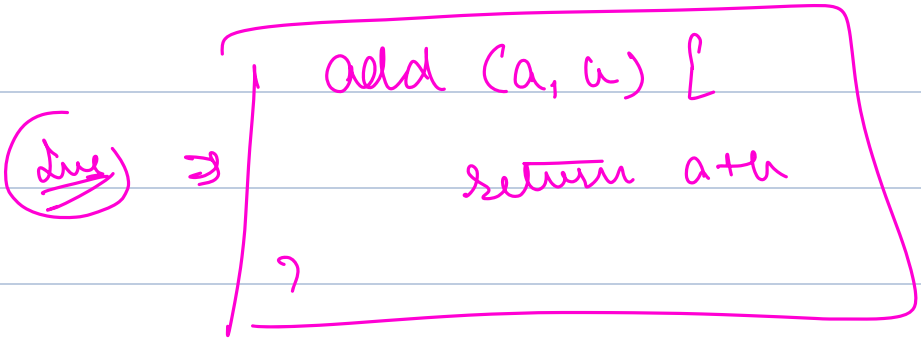
1. Intro to ML
2. Languages and ML → Vector Embedding
3. How to create vector embeddings
   - ↳ Word2Vec
   - ↳ BERT

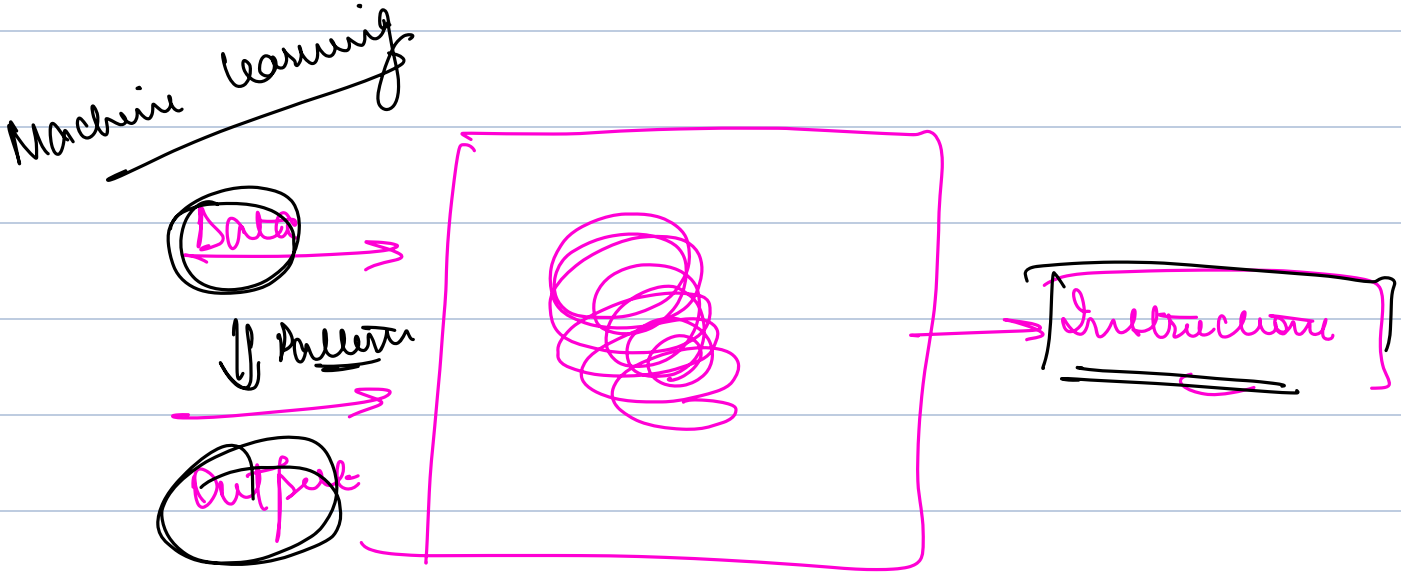
   (Transformer Architecture)
4. How LLMs work
5. RAG
6. Types of RAG → Text Embedding
7. Evaluation and adv topics.
8. Libraries
9. Retrieval Algos.

Intro to ML

Traditional programming

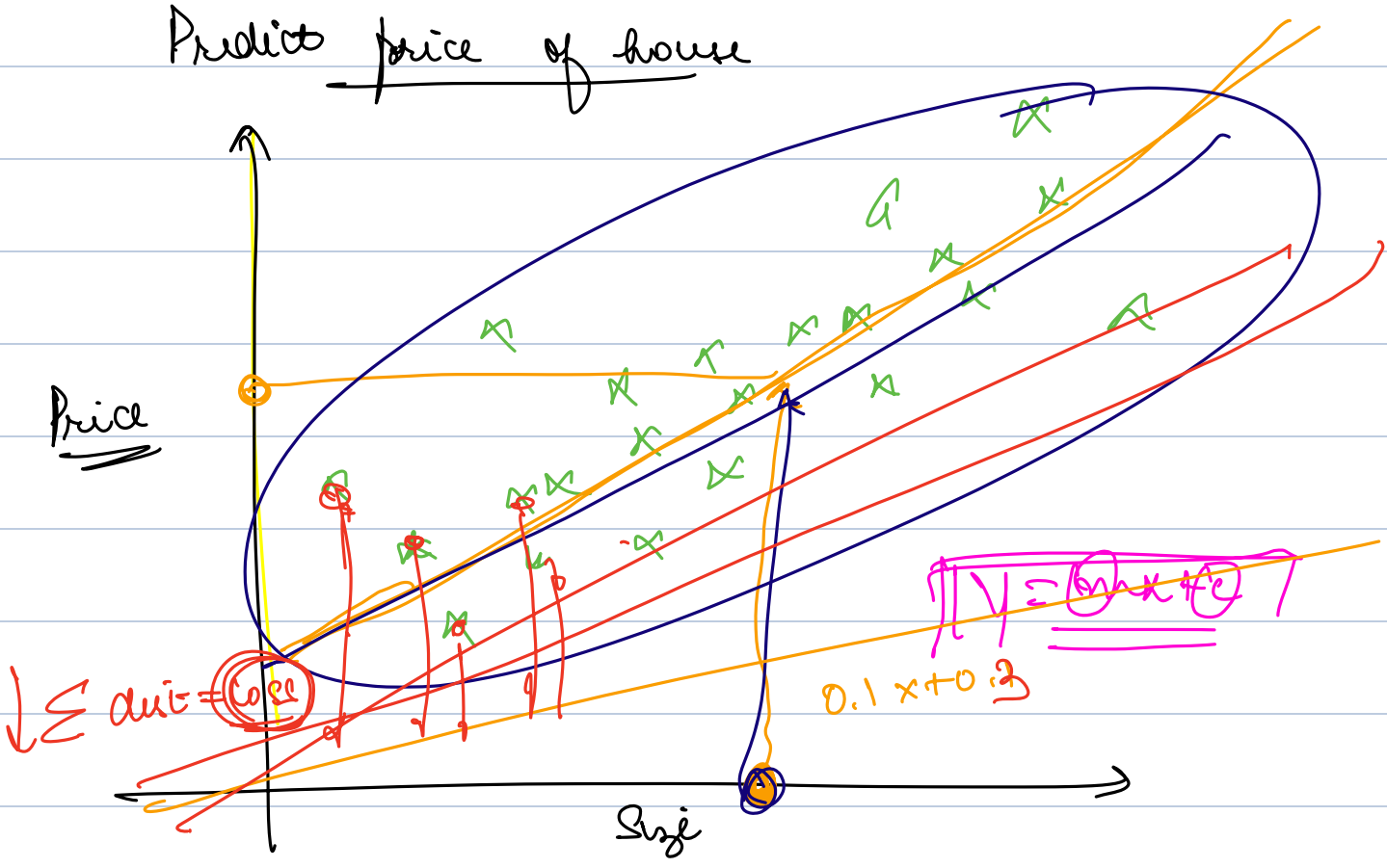






But How?

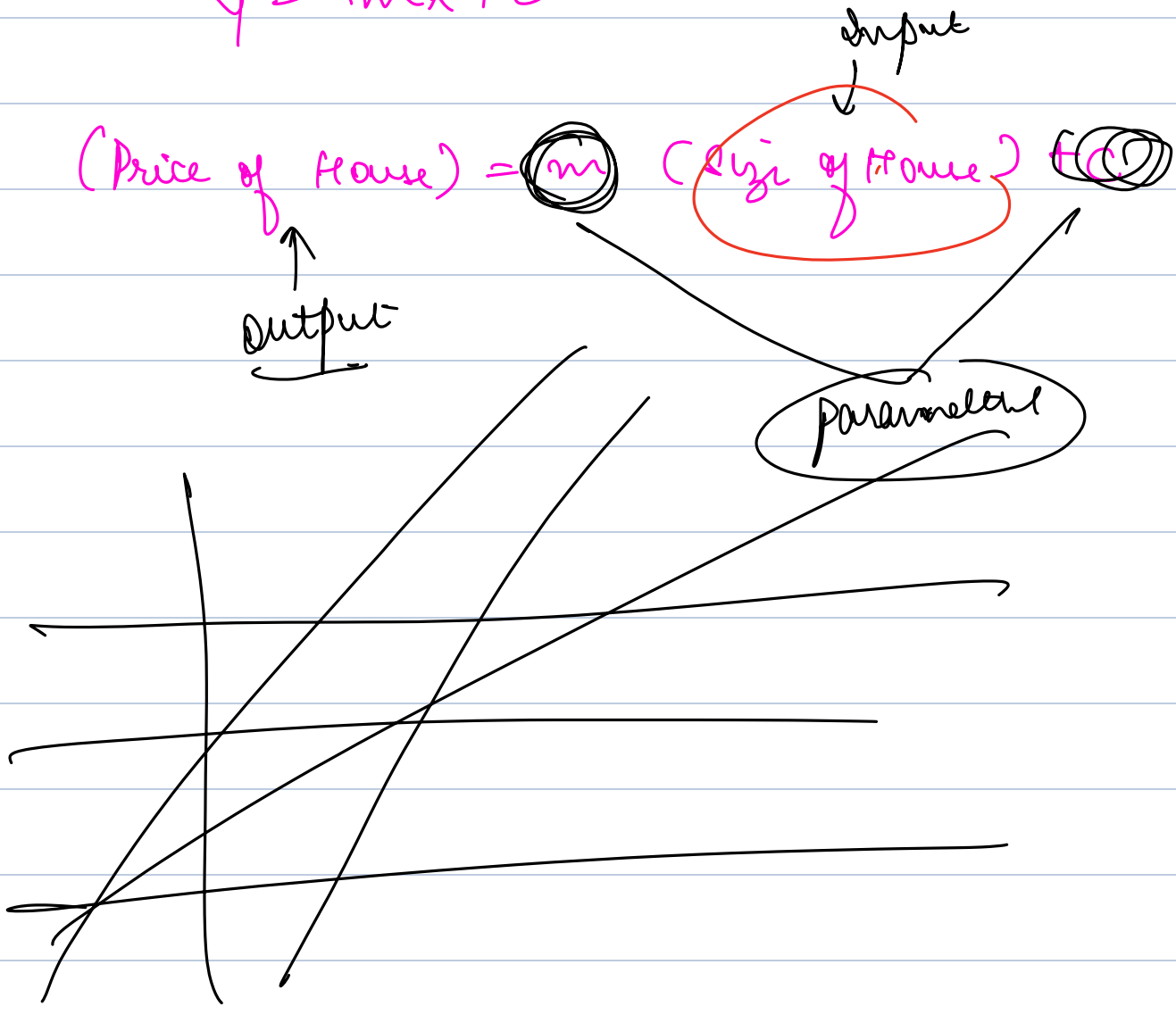
Predict price of house



Assumption behind ML : the world works on "desired pattern" we just have to figure the pattern

Assume the given case can be rep by a linear eq^{n} in x param

$$y = mx + c$$



$$y = ax_1 + bx_2 + c x_2^2 + \text{———}$$

Assume an eqn

$$y = mx + c$$

start by taking random value of these params

$$\Rightarrow \boxed{y = 0.1x + 0.1}$$

Iterate:
- till loss is minimized.
- Calculate how far is your current pattern from real data
- Based on that your change variables

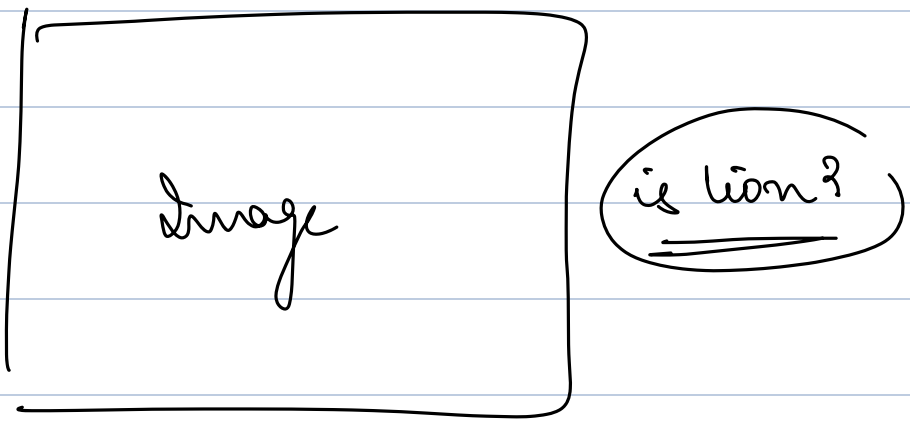
Gradient Descent

price = α_1 · Size + α_2 · #bedrooms
+ α_3 · # rooms

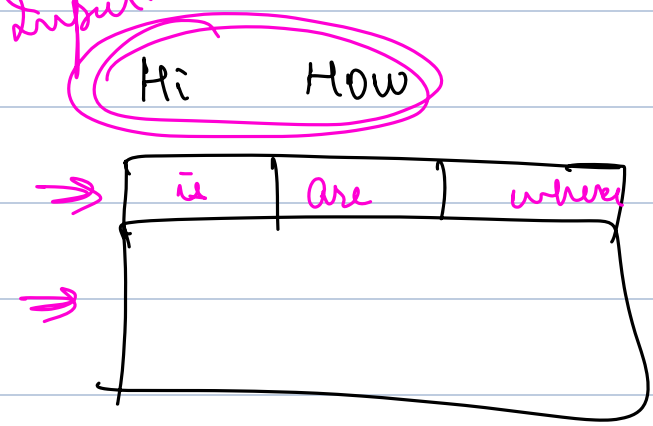
Inputs → Numbers

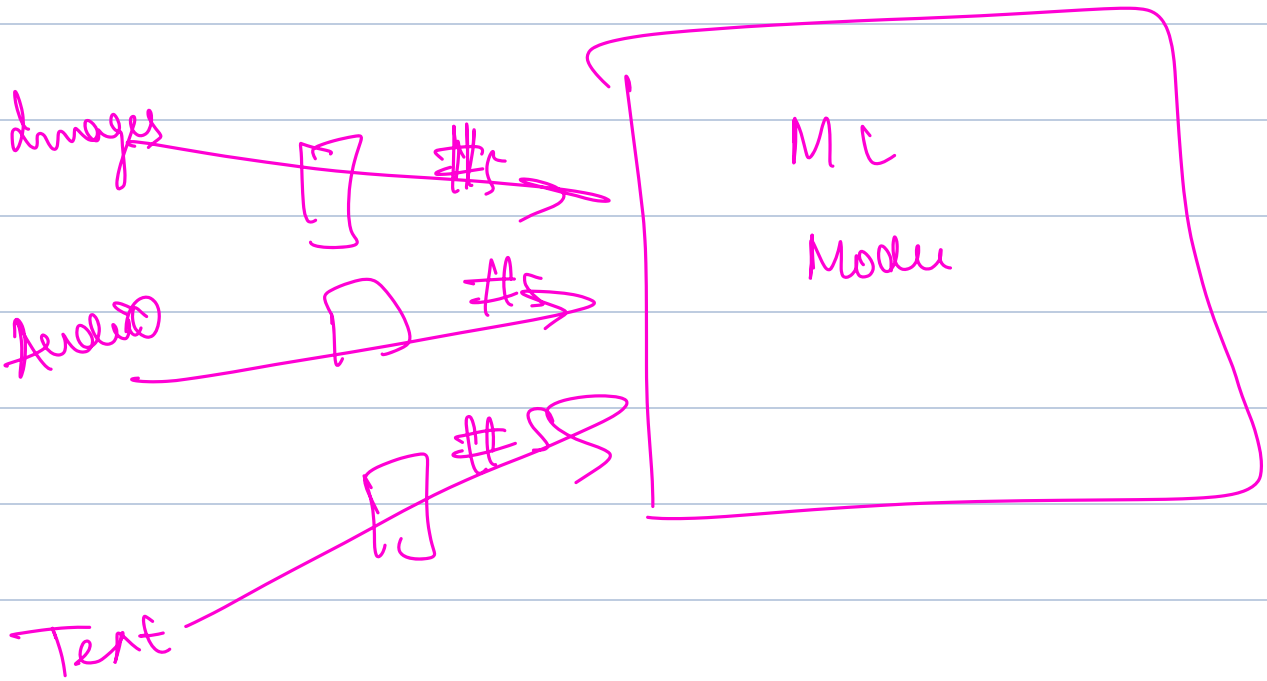
But!! Every kind of input for a machine prediction task may not be numeric.

Eg: Image



Input:





Text ⇒ Numbers

[NLP]

↳ Natural Language Processing

① Give every word a pos^n

Vocab ⇒ set of all words that I know about

(dictionary)

⇒ give rank to every word.

GPT3 ⇒ (52,227 words)

1 → a

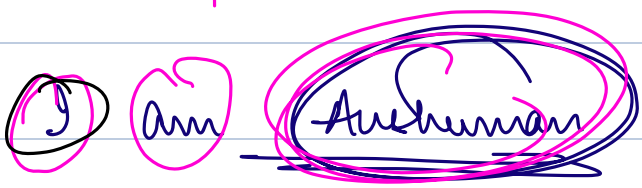
2 → abhishek

3 → aman

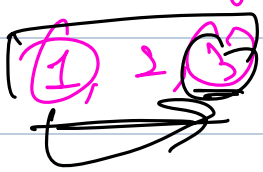
4 → appa

② have a appa = [37, 29, 1, 4]

5 → asset



Token By Token



[I, am, Arkumar, S, Kumar]

Input

↓

Tokens

↓

numeric rep for each

(2007)

215

But!!

There is no logical reasoning behind the #s I have for each word as.

So! our model may be
2. v.v. slow to learn
3. or may never learn at all.

⇒ our rep of words should have some meaning associated to them

Eg

$$Rep_{modi} - Rep_{india} = Rep_{Meloni} - Rep_{Italy}$$

$$Rep_{King} - Rep_{Queen} = Rep_{man} - Rep_{woman}$$

Can I do this?

Yes

Vector Embeddings

In vector embeddings, for each word in my dictionary, rather than figuring a numeric value, I try to figure a vector rep

Aman → 4

Aman →
$$\begin{bmatrix} 0.1 \\ 4.3 \\ 71.2 \\ -0.7 \\ -3 \\ 8.9 \end{bmatrix}$$

Physics

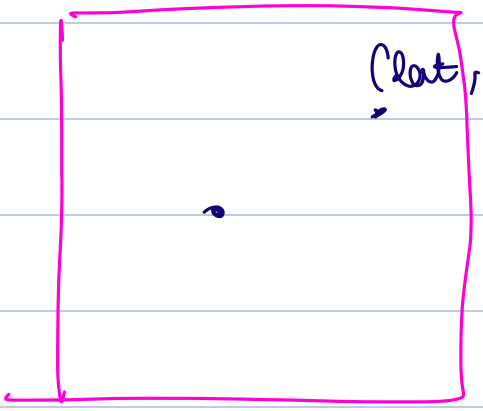
Scalar ⇒ Speed

Vector ⇒ Velocity

Matrix ⇒ 1D array

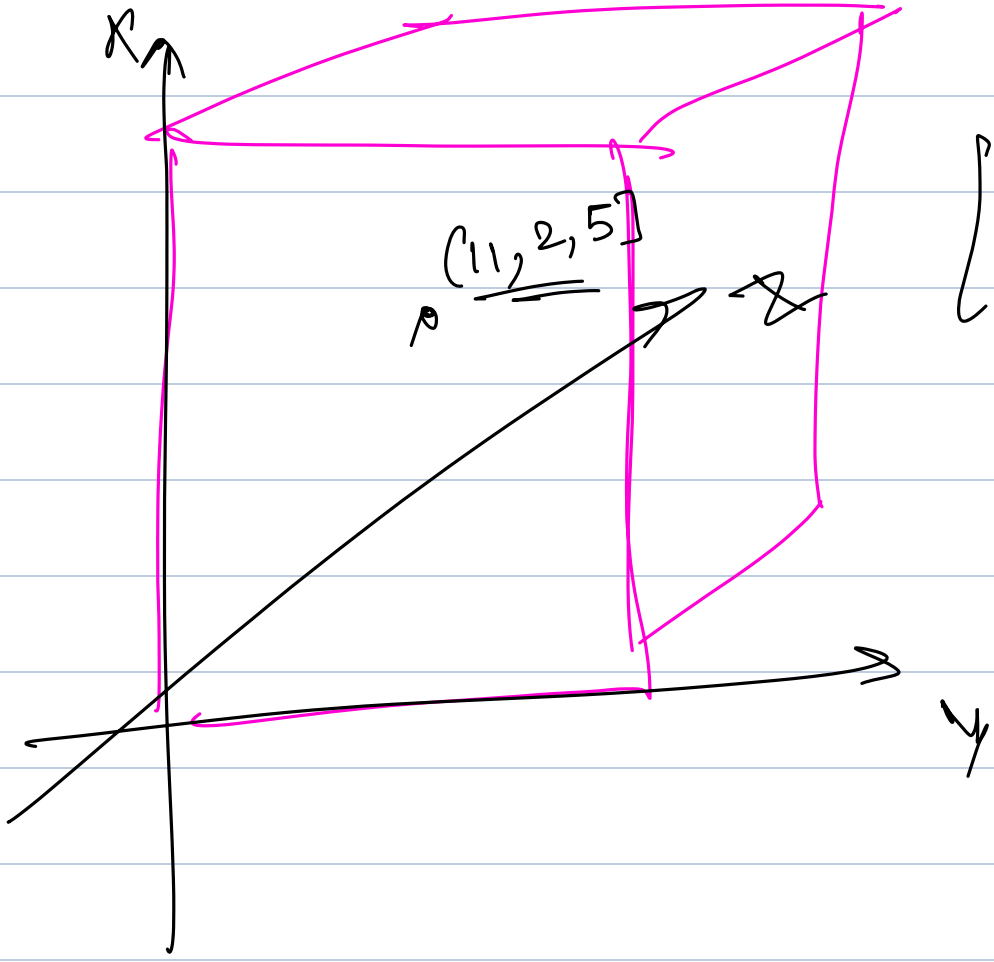
1D ⇒

2D ⇒



$$\begin{bmatrix} 3 \\ 4 \end{bmatrix}$$

3D

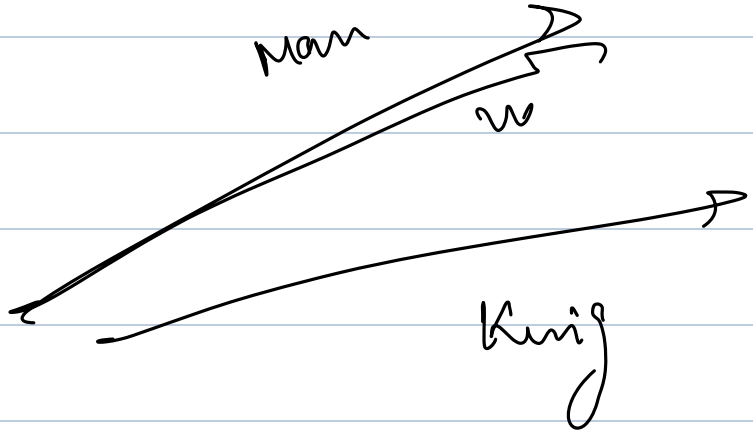


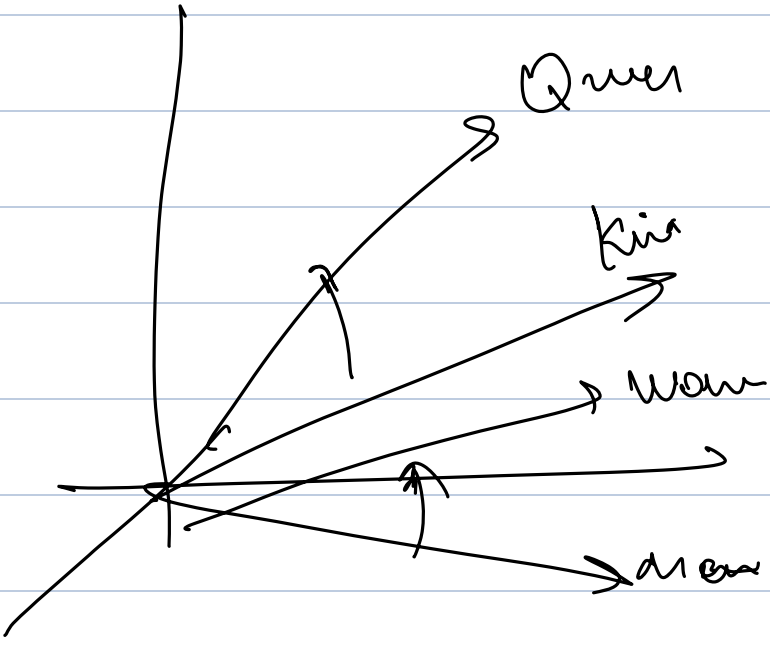
$$\begin{bmatrix} 11 \\ 2 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 \\ 7 \\ -3 \\ 2 \\ 9 \end{bmatrix}$$

GPT 3 12096 dimensions

[
0.1
7.3
-8.9
1.2
71.7
3.4
-91.9
⋮
]





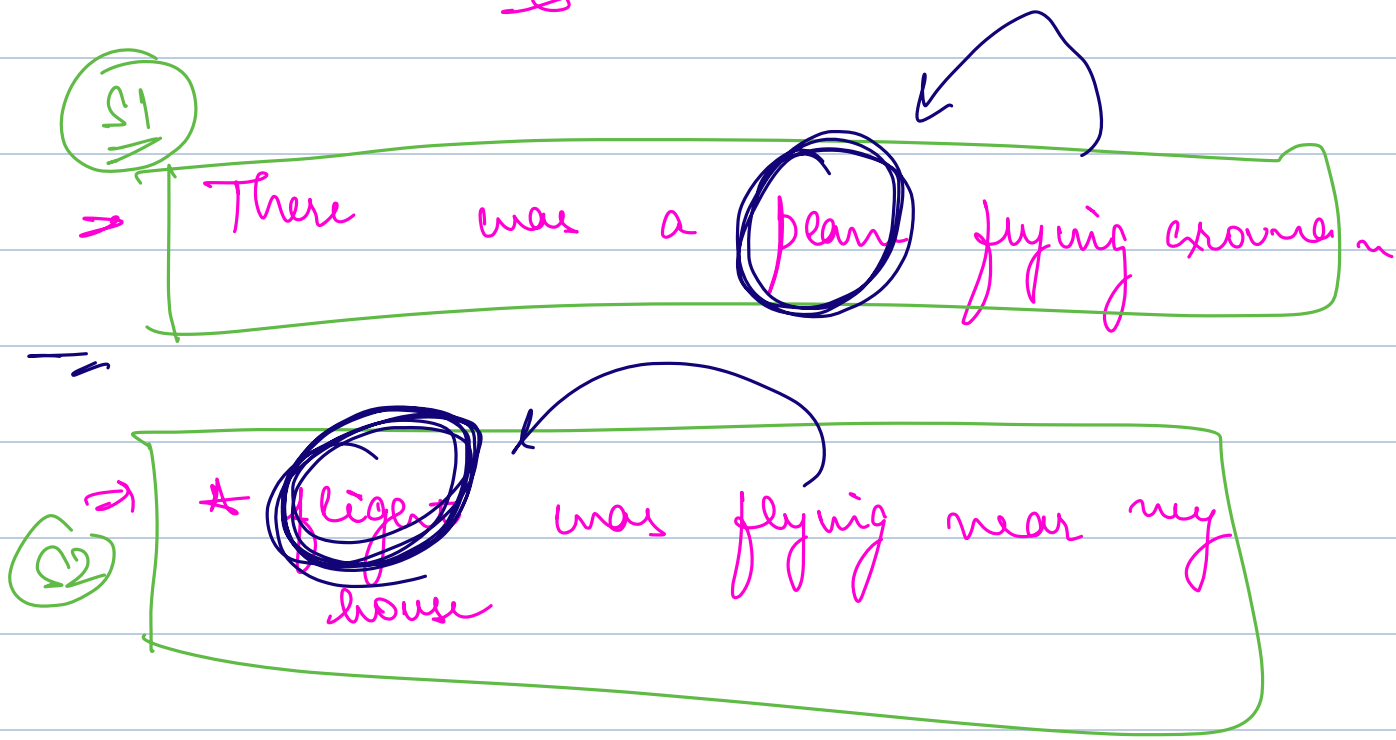
Vector Embed: Numeric Rep of words in my dictionary.

How? Cosine Similarity

= [distance $\vec{v}_1$ — $\vec{v}_2$]

men

boy



{ If 2 words have similar words around them, probably they are similar.

① Word 2 Vec

↳ CBOW (Continuous Bag of words)

↳ Skip Grams

CBOW ✗ Faster

whole wikipedia

There was a bird flying near me

The was flying a peane yesterday

The was an er wron

CBOW (

[There, was, __, a, plane]

↓ predict ?

flying ~~yesterday~~ Paper

Input

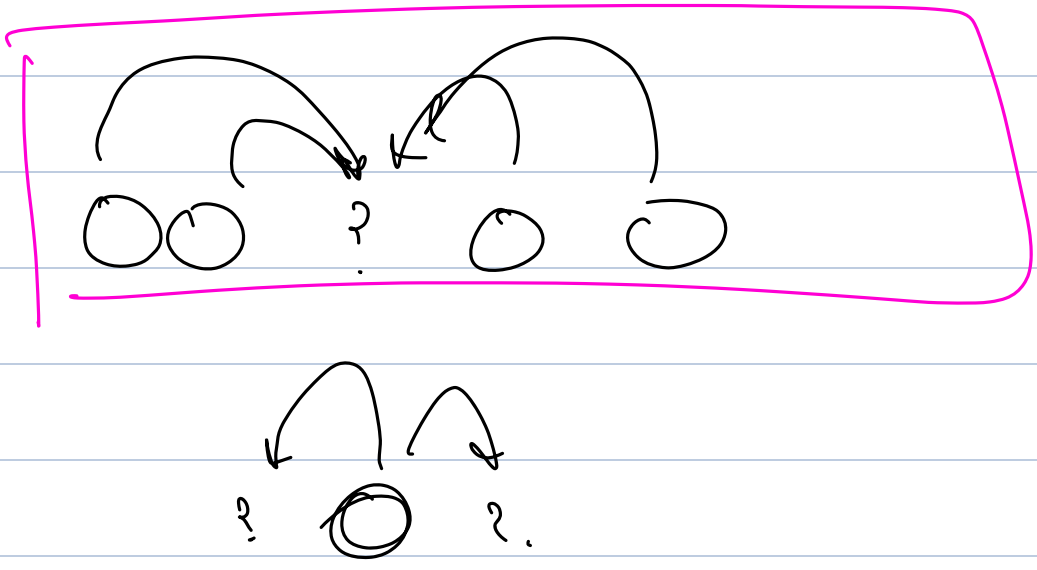
[There, was, _, _]

Op

_____

Skip Gram

predict the words around w



BERT =>

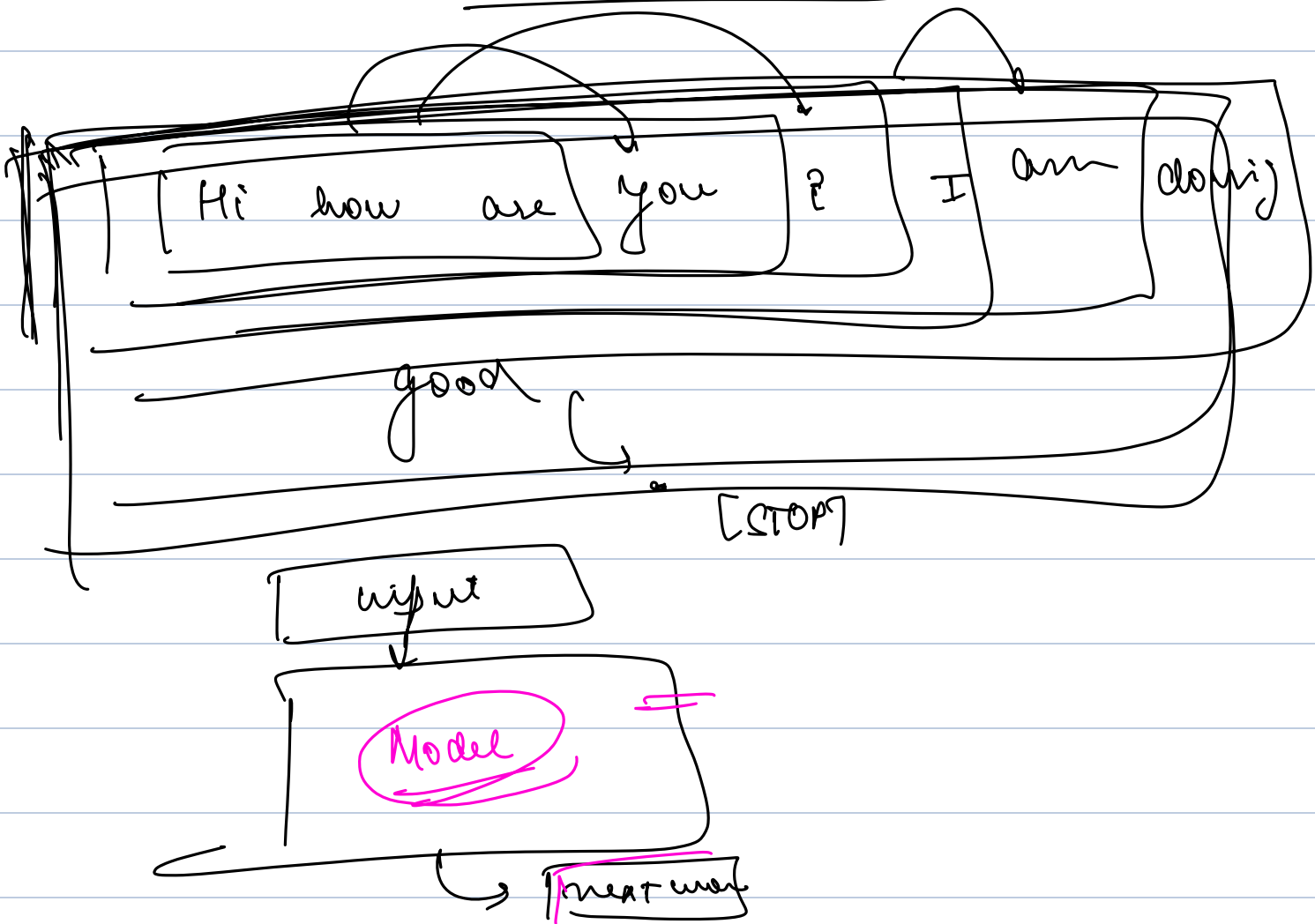
Intro to Transformer Architecture

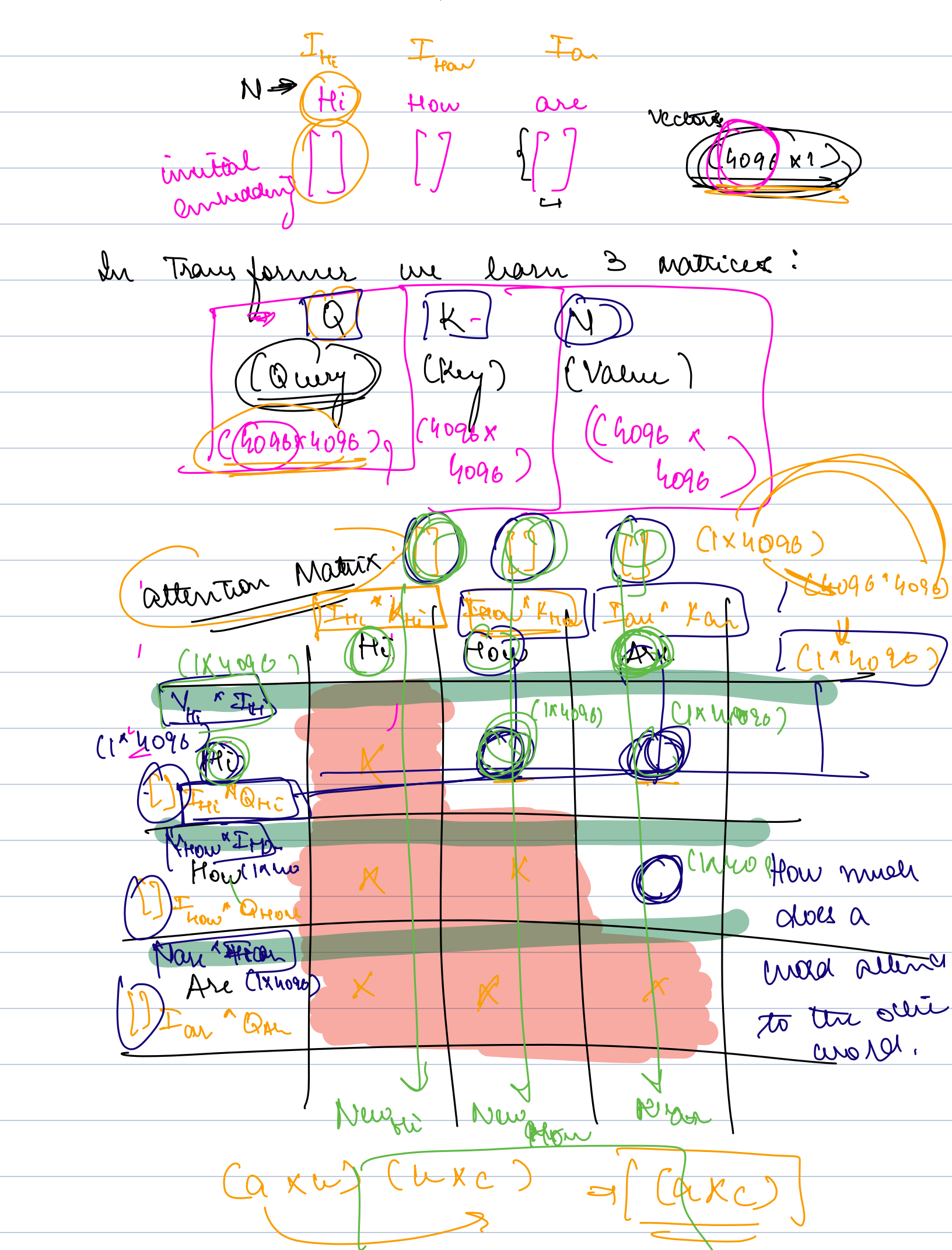
Assume an initial embedding for each token

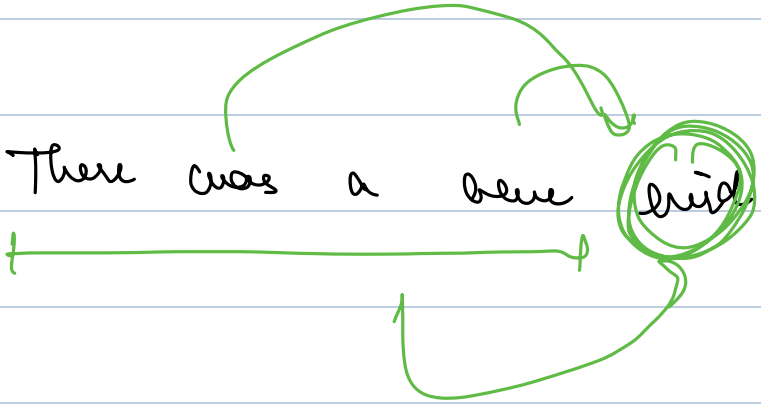
(we'll start by embeddings generated by word2vec)

GPT/LLM is nothing but

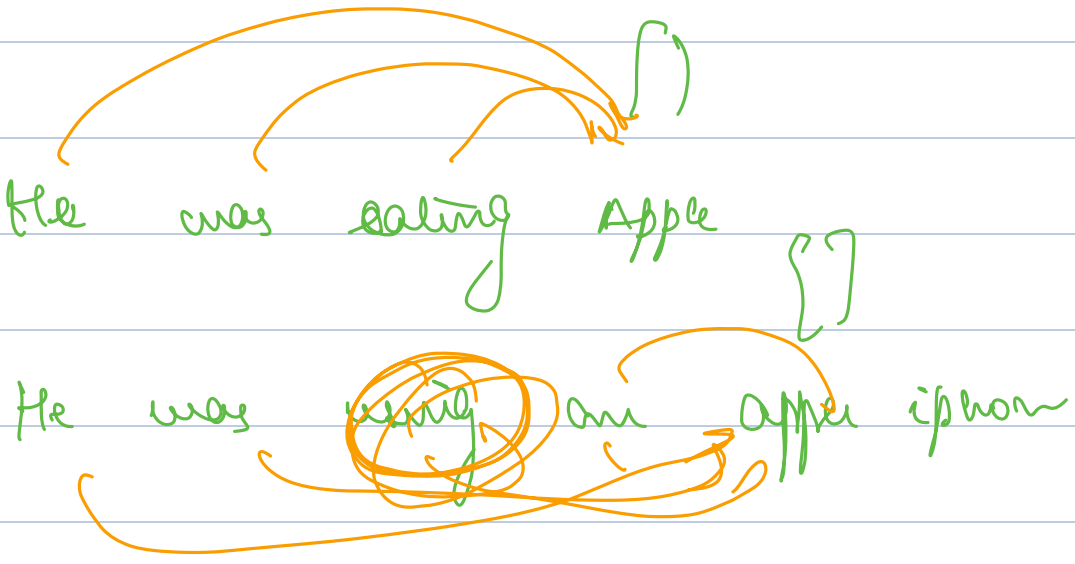
"Next word Predictor"




I_Hi
I_How
I_are
N
Hi
How
are
initial embedding
vectors
(4096 x 1)
In Transformer we learn 3 matrices:
Q
(Query)
(4096 x 4096)
K
(Key)
(4096 x 4096)
V
(Value)
(4096 x 4096)
attention Matrix
(1 x 4096)
(1 x 4096)
(4096 x 4096)
(1 x 4096)
(1 x 4096)
(1 x 4096)
(1 x 4096)
(1 x 4096)
How much does a word attend to the other words.
New_Hi
New_How
New_are
(a x b) (b x c) → (a x c)

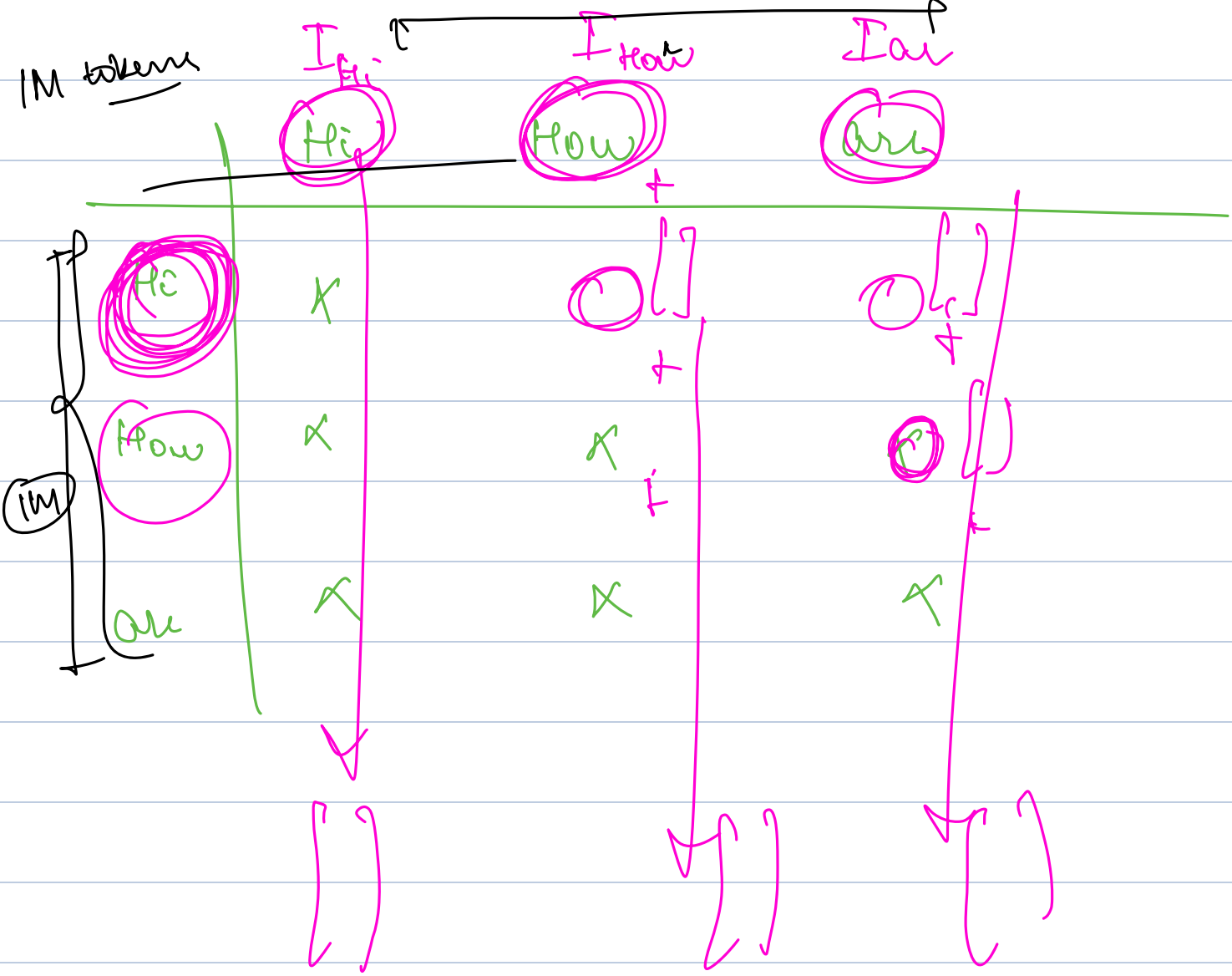


⇒ each preceding word may impact the meaning of future word

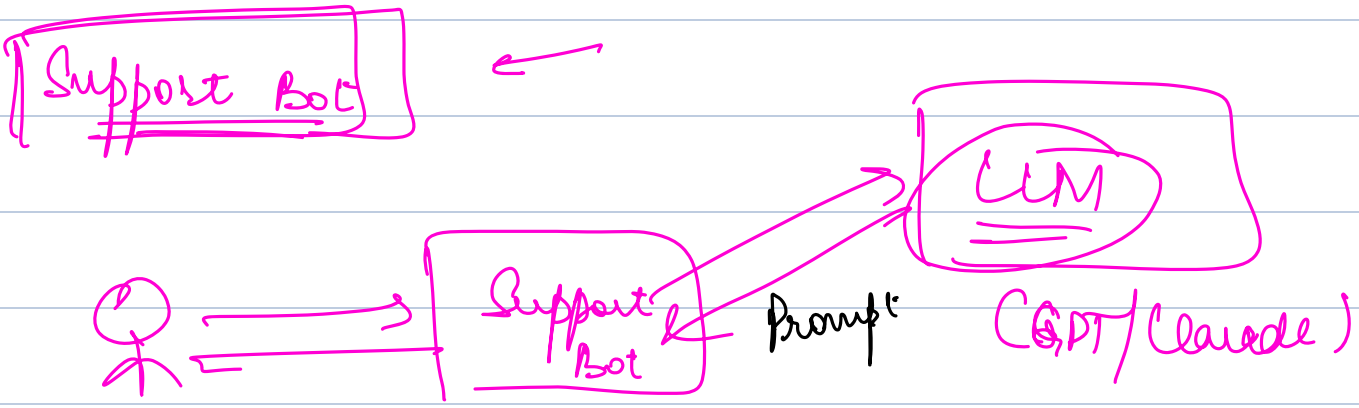


(1 × 4096) (1 × 4096) ≈

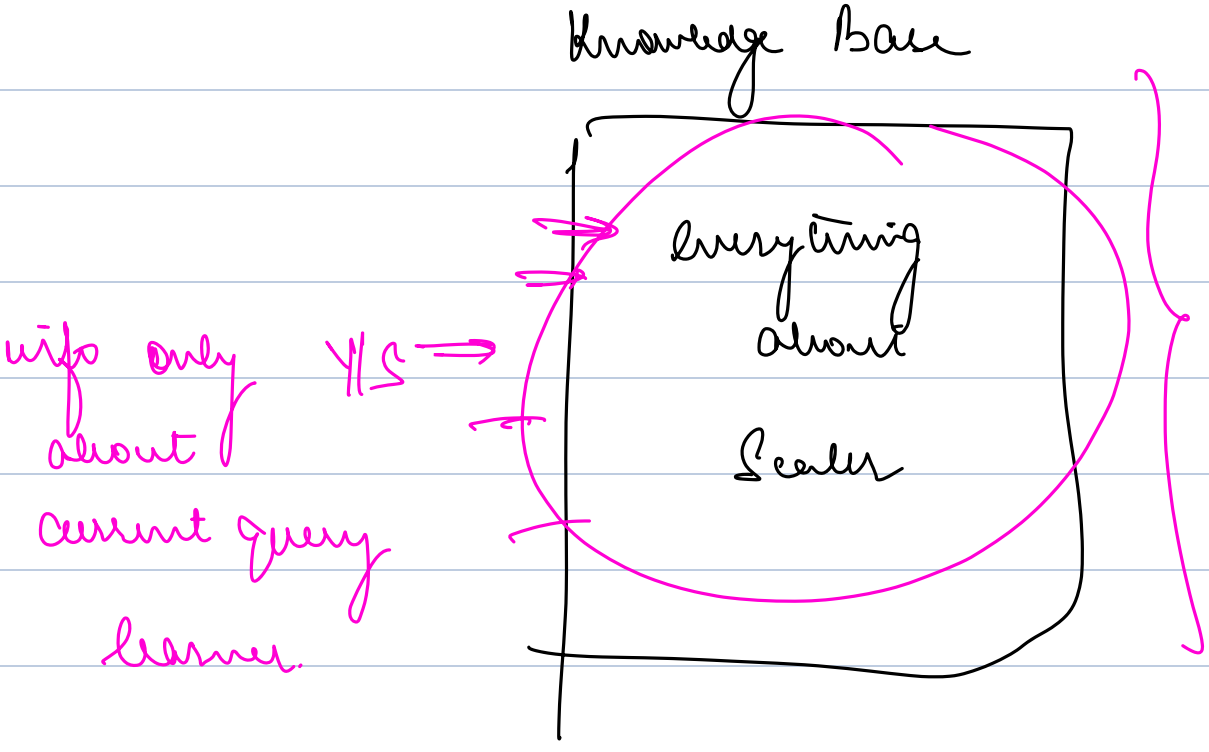
(4096 × 1)



RAG (Retrieval Augmented Generation)



You are



every LLM ⇒ context window

⇒ How much of input can it handle.

$10^6 \times 10^6$ ⇒ 10^{12}

4×10^{12} By ⇒ 4TB

Problem!

longer the prompt :

(1) More Costs

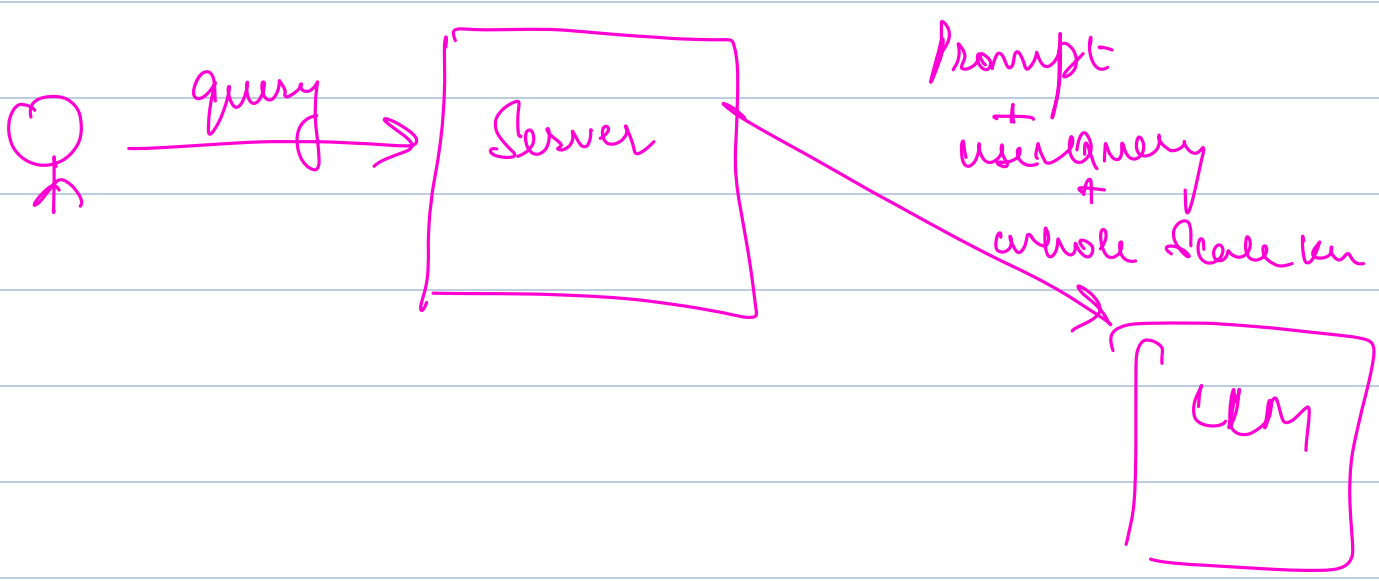
(2) Quality will also reduce.

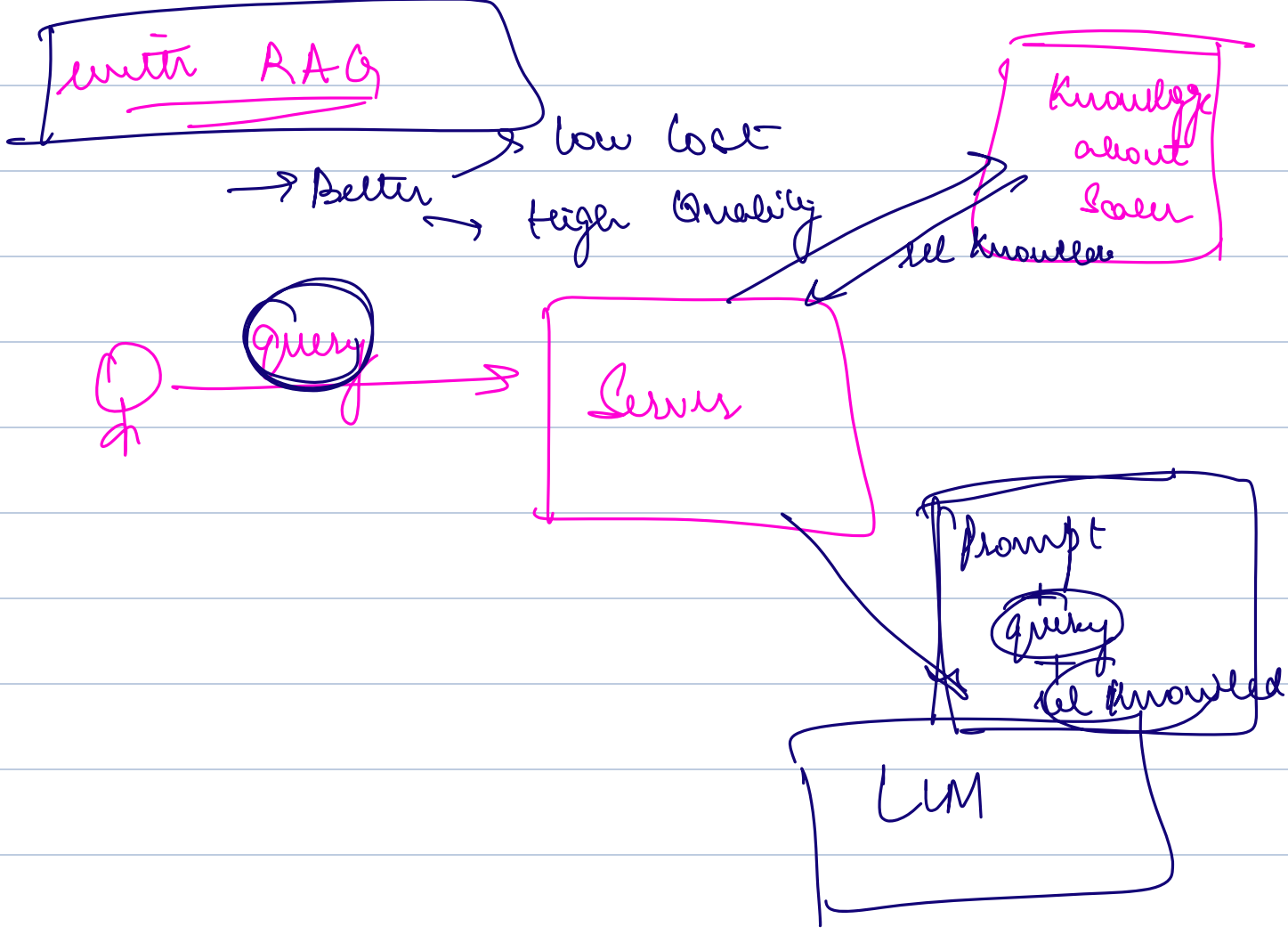
Use Case 1

⇒ give only relevant info to your model

↳ Save cost

↳ better quality.

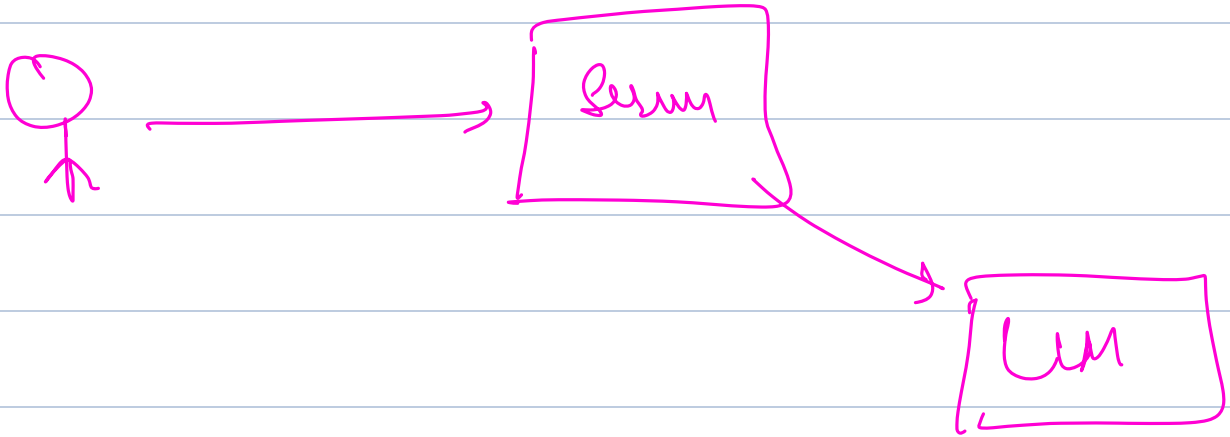


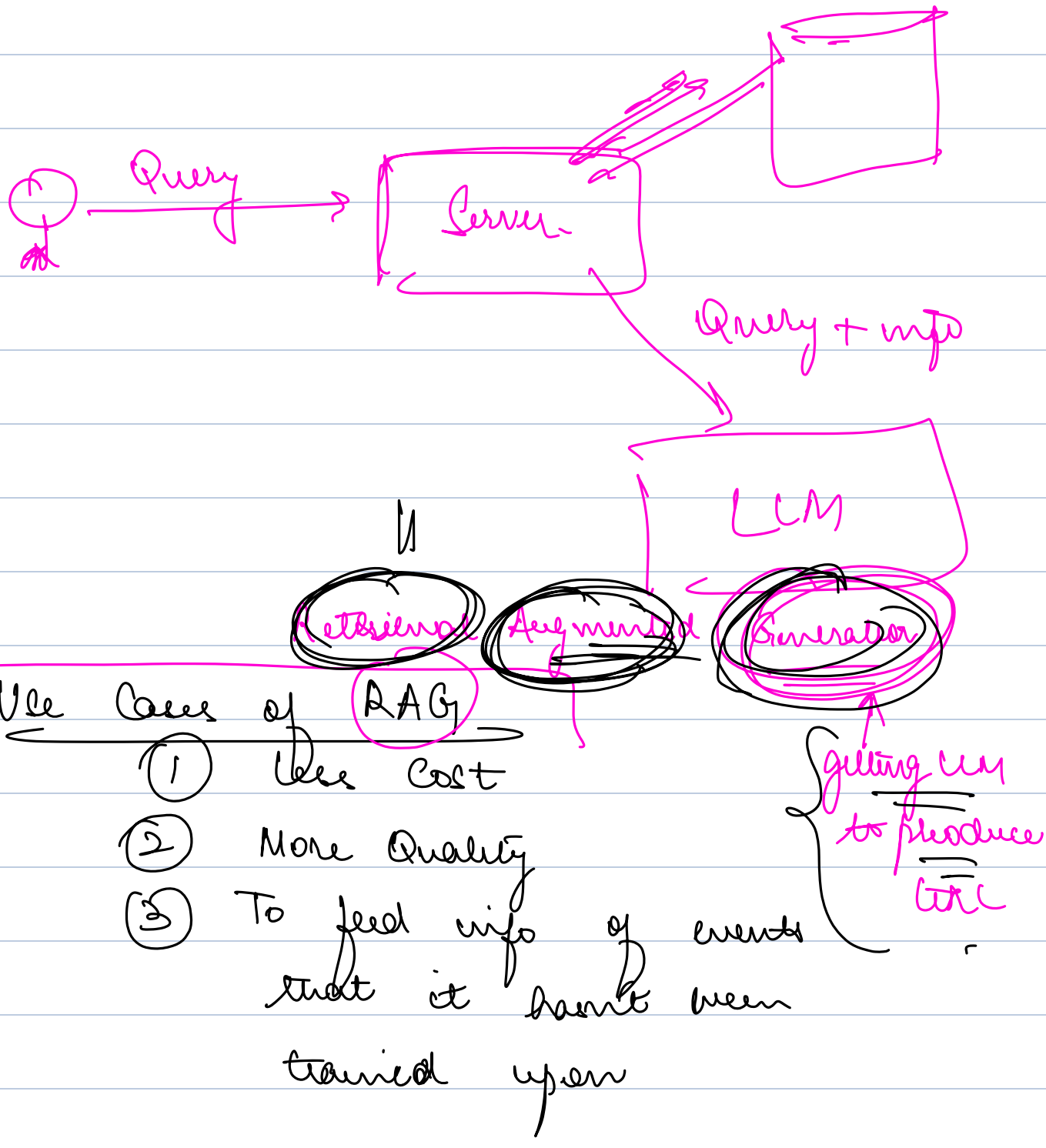


Issue 2

LLMs only know of info till the day were trained.

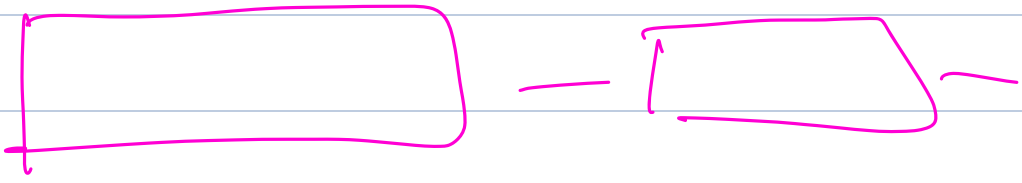
⇒ I might want to query about recent events



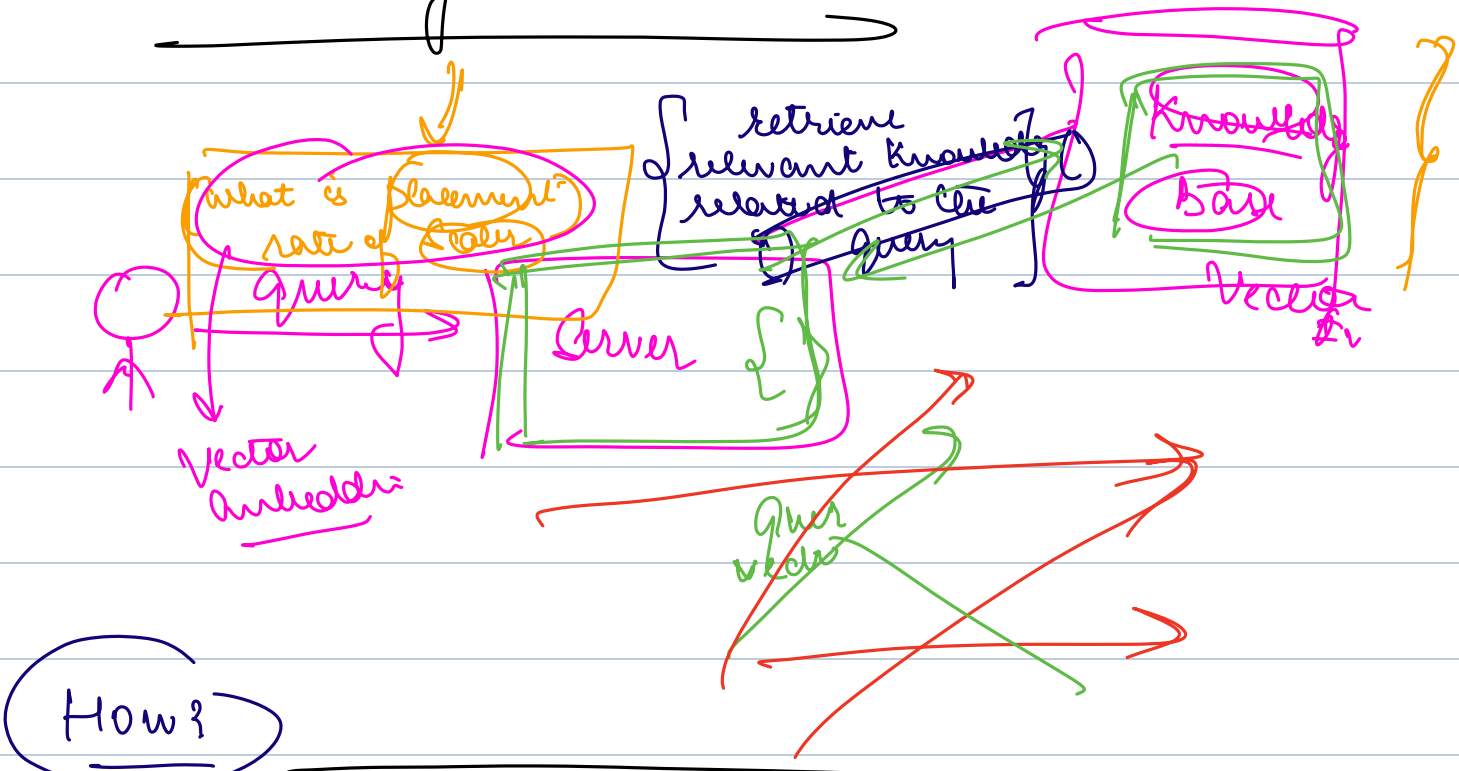


Use Cases of RAG

1. Less Cost
2. More Quality
3. To feed info of events that it hasn't been trained upon



How do you do RAG



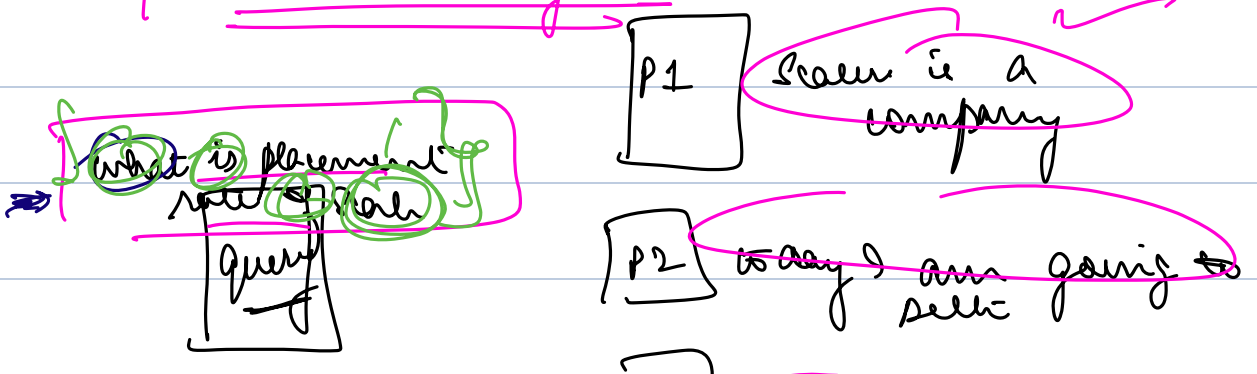
How?

1. Text Based Search
2. Embedding Based Search

Ans.

Text Based Search

TF - IDF algorithm



find which webpage is most relevant to my query.

(1) presence of words in my query in the page.

$\frac{2}{\text{Freq}}$ 1

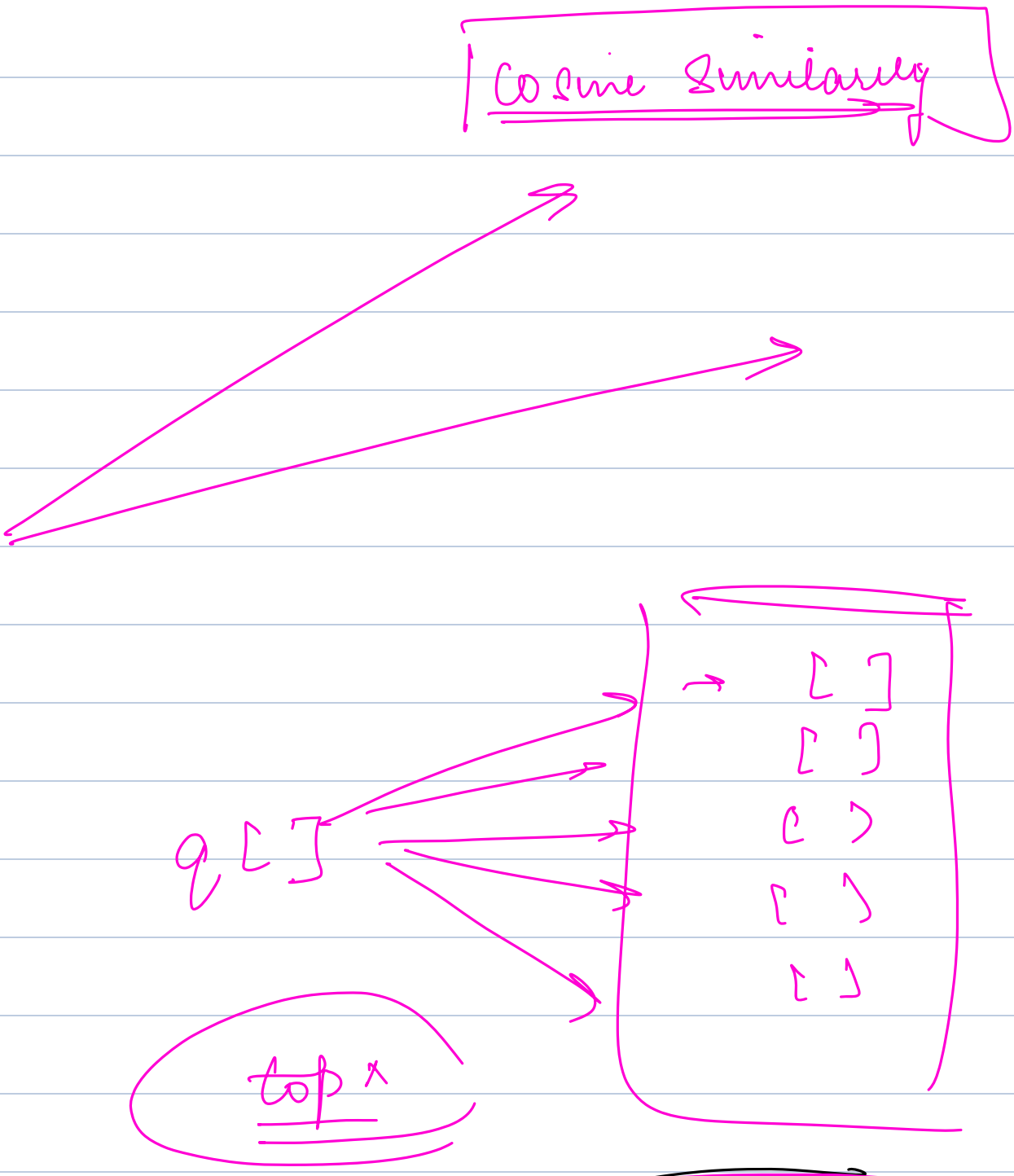
Term Frequency – Inverse Document Freq

Embeddings Based Search

(1) In DB I store BB embedding for every QnA pair.

(2) when a user sends a query I find closest vector embedding in DB

③ I give top 10 ~~of those~~ to LLM.



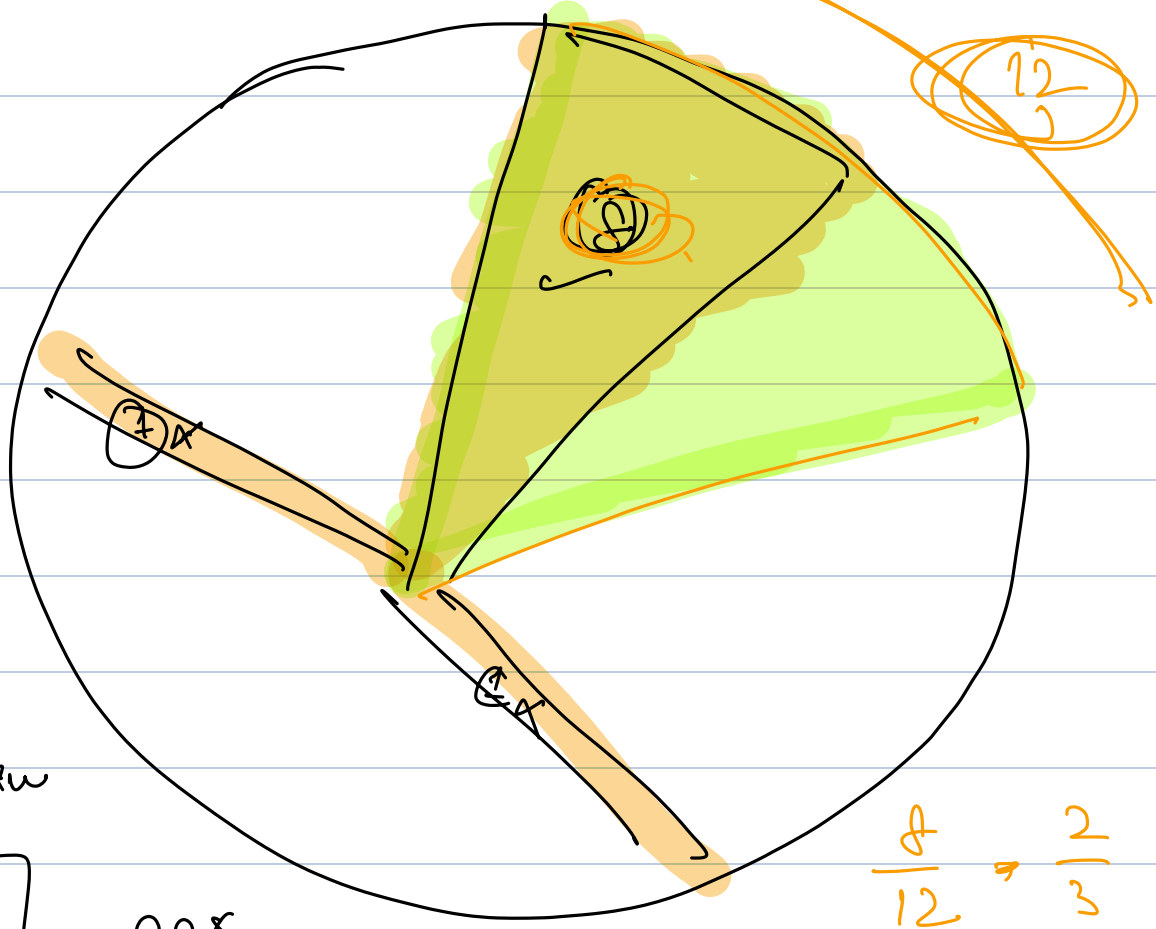
Optimization

How to eval if your RAG is good

1. Precision → % of doc that are returned that match my query
2. Recall → % of relevant docs that were ret

≈ 100% → 10 docs → 9 relevant → Precision → 90%

15 docs were relevant across internet → I got → 9/15



Precise

$\frac{8}{10} \rightarrow 80\%$

$\frac{8}{12} \rightarrow \frac{2}{3}$

66%.

Long Chain

Learn Index