GEN AI Basic to Advanced- Course Introduction

04 फरवरी 2025 12:33

Approach :

1) Will start from Basic and cover every thig you need to know
2) Combination of Theory and Practical
3) End to End App Development and Deployment
4) In our project we will capture how to initiate a project, plan it and complete using some of Agile Process

Pre-Requisites :

Willingness to Learn

Mandatory : Machine Learning Basic Concepts
Python Programing
Linear Algebra
Basics of Deep Learning

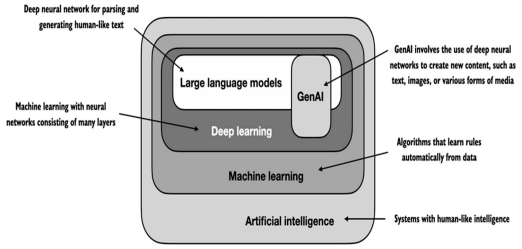
Books :

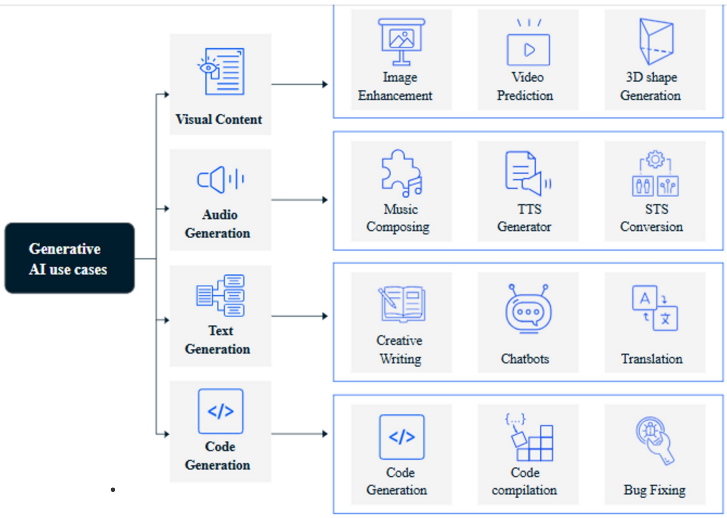
- Speech and Language Processing , Dan Jurafsky and James Martin
- Foundations of Statistical Natural Language Processing , Chris Maning and Hinrich Schtuze
- Build a Large Language model , Sebastian Raschka

Multiple Research Papers

What Is Gen AI ?

Generative AI, meaning a type of AI technology, that can create new content, such as text, audio, images, or videos. its applications and use cases are broad and varied, allowing to generate a story in the style of a specific author, producing realistic images of non-existent people, composing symphonies in the style of well-known composers, or creating a video clip from a simple text description



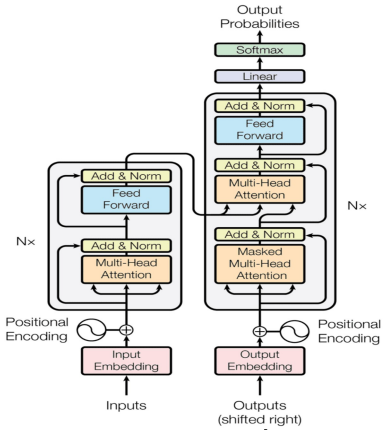


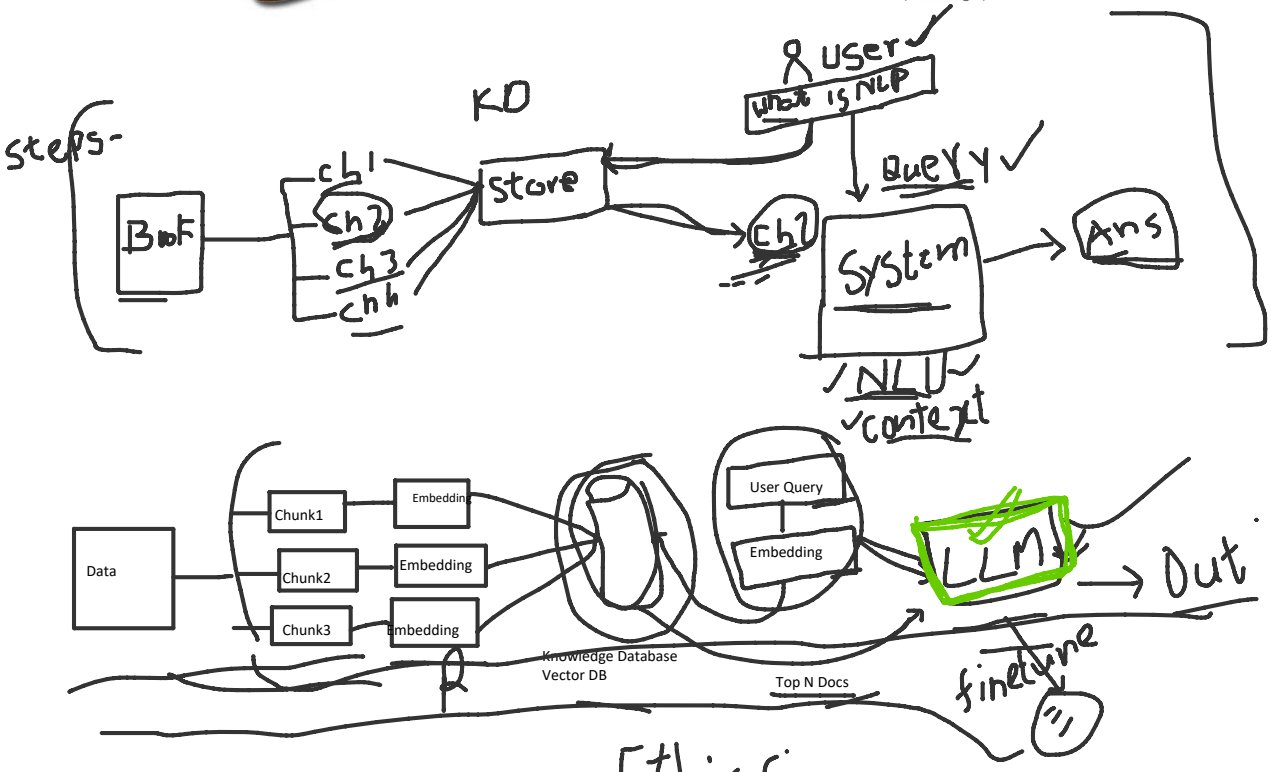
What is Large Language Model (LLMs) ?

Neural Network Designed to Understand, generate and respond to human like text.

These Models are deep Neural Networks trained on massive amount of text data.

What Makes LLMs so Good?





Ethics

Course Content :

1) Introduction
2) Quick Recap of Linear Algebra
3) Build your own LLM Models
    a. Data Processing
    b. Transformer Architecture
    c. Training and Evaluation
    d. Save Parameters
4) Retrieval Augmented Generation (RAG)
5) Fine Tuning
6) Lang chain
7) LLM OPS
8) End to End Projects

Session 2 :

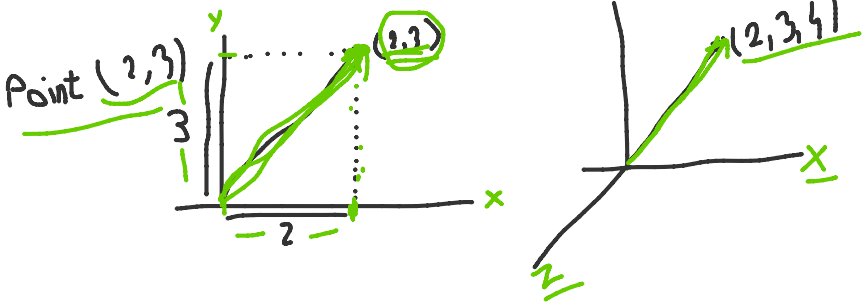
Gen AI End to End course

Today's Agenda

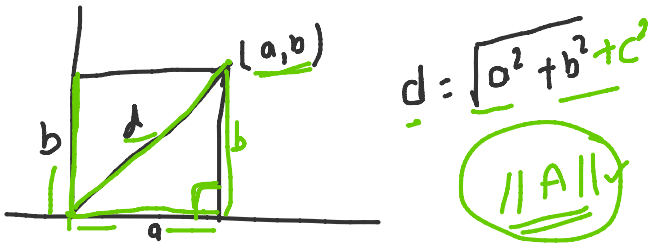
1) Recap Basics of Vectors and Matrix
2) Start Building Own LLM Model

Linear Algebra

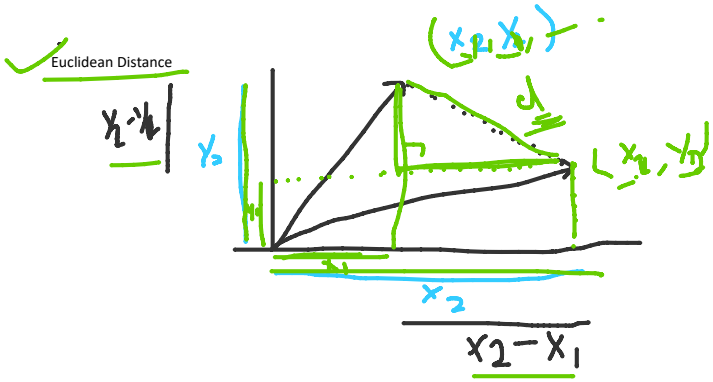
Vectors



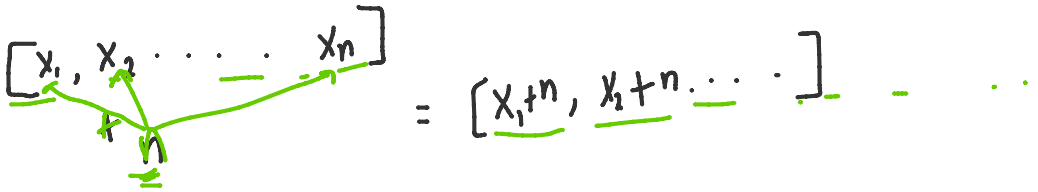
DISTNCE of a Vector from Origin



Euclidean Distance



Scaler Addition



Dot Product

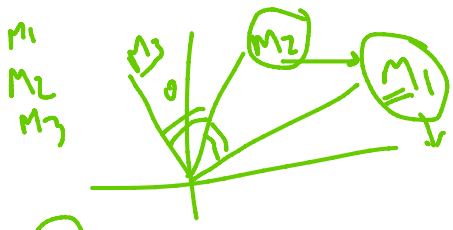
If we have two vectors:

- $A = (A_1, A_2, ..., A_n)$
- $B = (B_1, B_2, ..., B_n)$

The dot product of **A** and **B**, denoted by **A · B**, is calculated by multiplying corresponding components and then adding those products together:

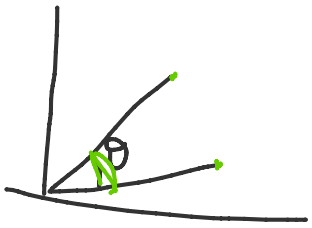
The dot product of **A** and **B**, denoted by **A · B**, is calculated by multiplying corresponding components and then adding those products together:

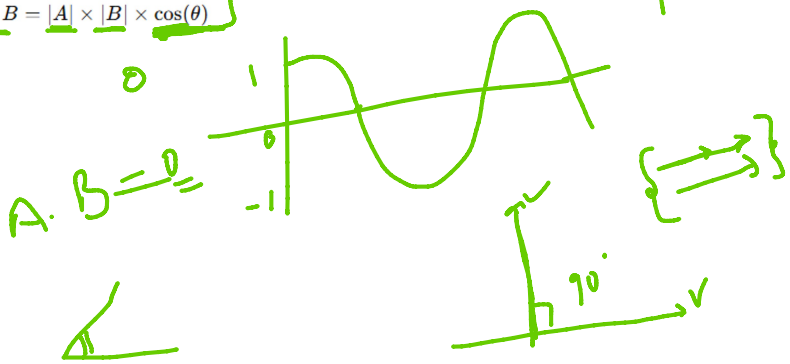
$$A \cdot B = A_1 \times B_1 + A_2 \times B_2 + \cdots + A_n \times B_n$$



- The dot product also has a geometric interpretation: it's related to the angle **θ** between the two vectors.

$$A \cdot B = |A| \times |B| \times \cos(\theta)$$





- Matrix

A **matrix** is a rectangular array of numbers or other mathematical objects arranged in rows and columns. It's a way to organize and represent data in a compact form, and it's used extensively in various fields like mathematics, physics, computer science, and engineering.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

1. **Addition**: You can add two matrices if they have the same size (i.e., the same number of rows and columns). Add the corresponding elements.

   Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

Matrix Multiplication (Dot Product of Matrices)

Matrix **A** (2x3 matrix):

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Matrix **B** (3x2 matrix):

$$B = \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$$

Matrix multiplication process:

1. Multiply the **first row of A** with the **first column of B** to get the first element of the result matrix:

$$(1 \times 7) + (2 \times 9) + (3 \times 11) = 7 + 18 + 33 = 58$$

2. Multiply the **first row of A** with the **second column of B** to get the second element:

$$(1 \times 8) + (2 \times 10) + (3 \times 12) = 8 + 20 + 36 = 64$$

3. Multiply the **second row of A** with the **first column of B** to get the third element:

$$(4 \times 7) + (5 \times 9) + (6 \times 11) = 28 + 45 + 66 = 139$$

4. Multiply the **second row of A** with the **second column of B** to get the fourth element:

$$(4 \times 8) + (5 \times 10) + (6 \times 12) = 32 + 50 + 72 = 154$$

Matrix multiplication process:

1. Multiply the **first row of A** with the **first column of B** to get the first element of the result matrix:

$$(1 \times 7) + (2 \times 9) + (3 \times 11) = 7 + 18 + 33 = 58$$

2. Multiply the **first row of A** with the **second column of B** to get the second element:

$$(1 \times 8) + (2 \times 10) + (3 \times 12) = 8 + 20 + 36 = 64$$

3. Multiply the **second row of A** with the **first column of B** to get the third element:

$$(4 \times 7) + (5 \times 9) + (6 \times 11) = 28 + 45 + 66 = 139$$

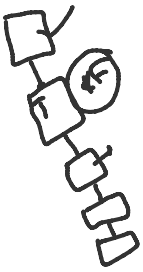
4. Multiply the **second row of A** with the **second column of B** to get the fourth element:

$$(4 \times 8) + (5 \times 10) + (6 \times 12) = 32 + 50 + 72 = 154$$

So, the resulting matrix **C** (which is a **2x2** matrix) is:

$$C = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Lets Start with LLM



Building on LLM

Foundational Model

Data Preparation

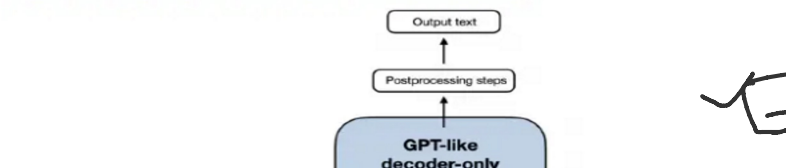
Attention Mechanism

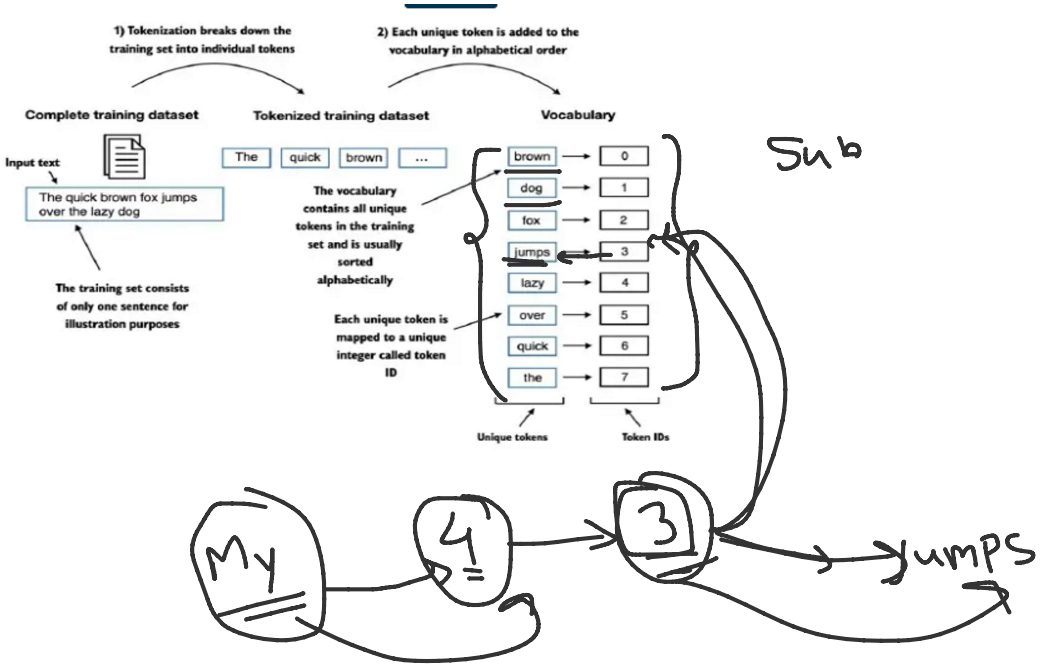
LLM Architecture

Train The Model

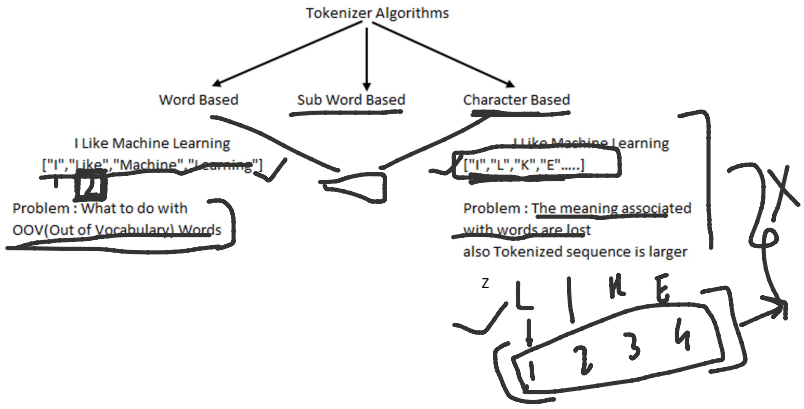
Model Evaluation

Load Pretrain Model





Today's Agenda:
Word Tokenization and challenges with it
Byte Pair Encoding



Sub Word Tokenization

Rule 1 : Do not split Frequent words into smaller sub words
Rule 2 : Split the rare words into smaller and meaningful sub words

Example : Boy should not split, boys can be split in Boy and s

BYTE Pair Encoding :

Is a sub word tokenization algorithm

BPE Algorithm : Most common pair of consecutive bytes of data is replaced with a byte that does not occur in data.

Research Paper : [PDF] A new algorithm for data compression | Semantic Scholar

Explain BPE :

Original Data : aaabdaaabac

1) The Byte pair "aa" occurs the most and we will replace it with "z" as "z" is not present in data.
2) Compressed data : zabdzabac

3) The next most common byte pair is "ab". We will replace it with "x"
4) Compressed data : zxdzxac
5) Only ac is the byte pair left but it appears once so don't replace it
6) We can replace "zx" with "y",
7) Compressed data : ydyac

How BPE Algorithm is used in LLMs

BPE ensures that most common words in the vocabulary are represented as a single token, while rare words are broken down into two or more sub words

Lets take an example : { "old" : 7 , "older" : 3 , "finest" : 9 , "lowest" : 4}

Processing : we need to add end token "</w> at the end of each word.

{ "old</w>" : 7 , "older</w>" : 3 , "finest</w>" : 9 , "lowest</w>" : 4}

Now split words into characters and count their frequency :

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13
11	t	13
12	w	4

Next Step in BPE is to look for the most frequent pairing : merge them and perform same iteration again and again until we reach the token limit or iteration limit

Iteration 1 : Start with second most common token "e". Most common byte pair with e is "es"

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0
11	t	13
12	w	4
13	es	9+4=13

Iteration 2 : merge token "es" and "t" as they have appeared 13 times in our dataset.

Number	Token	Frequency
1	</w>	23
2	o	14
3	l	14
4	d	10
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0
11	t	13-13=0
12	w	4
13	es	9+4=13-13=0
14	est	9+4=13

Iteration 3 :

Now let us look at </w> token. We see "est</w>" appeared 13 times.

Number	Token	Frequency
1	</w>	23-13=10
2	o	14
3	l	14
4	d	10
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0
11	t	13-13=0
12	w	4
13	es	9+4=13-13=0
14	est	9+4=13-13=0
15	est</w>	13

This will help understand the diff in estimate and highest

Iteration 4: "o" and "l" occurs 10 times.

Number	Token	Frequency
1	</w>	23-13=10
2	o	14-10 =4
3	l	14-10=4
4	d	10
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0
11	t	13-13=0
12	w	4
13	es	9+4=13-13=0
14	est	9+4=13-13=0
15	est</w>	13
16	ol	10

Iteration 5: "ol" and "d" occurs 10 times.

Number	Token	Frequency
1	</w>	23-13=10
2	o	14-10 =4
3	l	14-10=4
4	d	10-10=0
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0
11	t	13-13=0
12	w	4
13	es	9+4=13-13=0
14	est	9+4=13-13=0
15	est</w>	13
16	ol	10-10=0
17	old	10

"f","l" and "n" appears only in one word. So will not token them separately.

Now lets remove the token with zero frequency

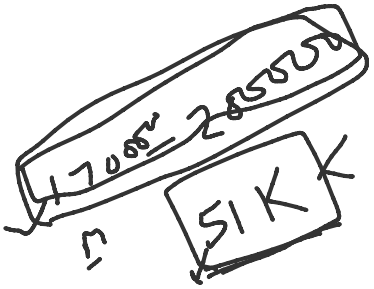
Number	Token	Frequency
1	</w>	23-13=10
2	o	14-10 =4
3	l	14-10=4
4	d	10-10=0
5	e	16-13=3
6	r	3
7	f	9
8	i	9
9	n	9
10	s	13-13=0

Number	Token	Frequency
1	</w>	23-13=10
2	o	14-10 =4
3	l	14-10=4
4	e	16-13=3
5	r	3
6	f	9
7	i	9
8	n	9
9	w	4

fine

8	i	9
9	n	9
10	s	13-13=0
11	t	13-13=0
12	w	4
13	es	9+4=13-13=0
14	est	9+4=13-13=0
15	est</w>	13
16	ol	10-10=0
17	old	10

7	i	9
8	n	9
9	w	4
10	est</w>	13
11	old	10



This list of 11 tokens will serve as vocabulary.

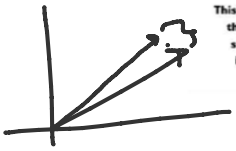
The stopping criteria can either be token count or number of iteration.

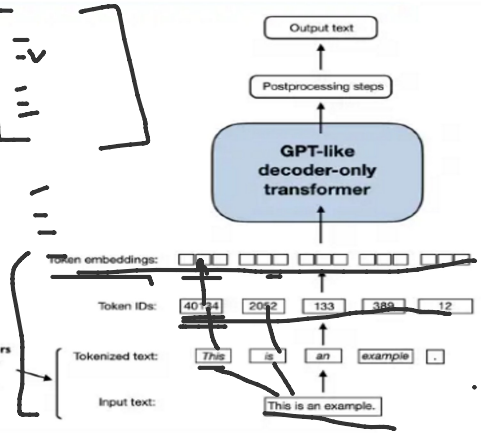
Token Embedding

How can we represent text into numbers

Can we assign random numbers to each words?

Dog --------> 20
Cat ---------->10
King ---------->-30
Kitten -------->-15



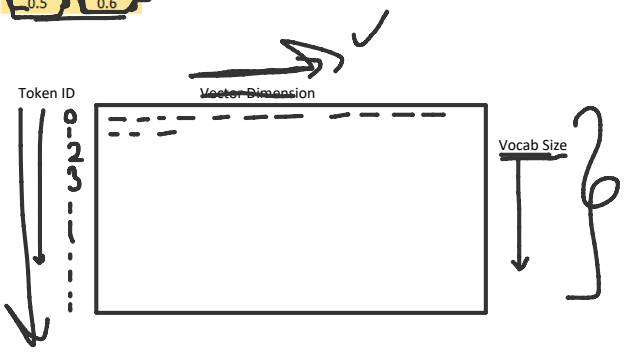


What if we convert based on features?

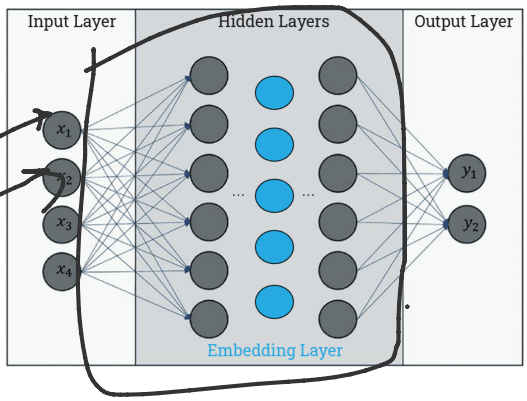
Example :

Features	King	Queen	Dog	Cat	Apple	Stone
Human	30	28	1	2	0.5	1.5
Has two legs	25	30	2	2	3	1
has four legs	1	1	35	29	0	-2
is eatable	1	1.5	2	2.3	29	28
is a pet	2	2.1	33	31	0.5	0.6

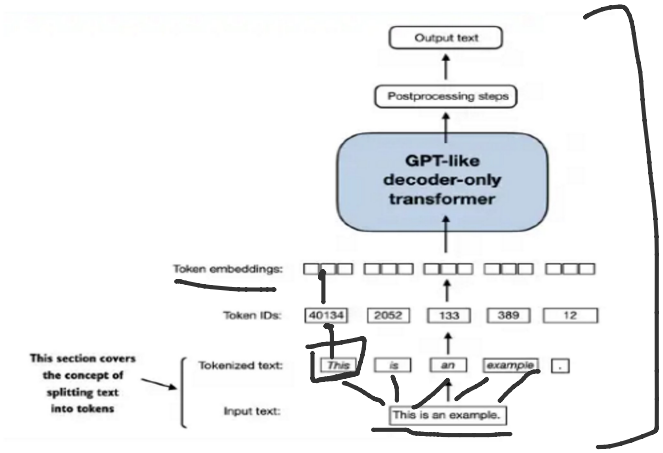
Token	Token ID	Embedding
Token1	1	[1,2,3,4,5]
Token2	2	[2,2,3,4,5]
Token3	3	[3,2,3,4,5]
Token4	4	[4,2,3,4,5]
Token5	5	[5,2,3,4,5]
Token6	6	[6,2,3,4,5]
Token7	7	[7,2,3,4,5]
Token8	8	[8,2,3,4,5]
Token9	9	[9,2,3,4,5]
Token10	10	[10,2,3,4,5]
Token11	11	[11,2,3,4,5]



How to Get These features and Embedding values



Agenda :
Create Input-Target Pair



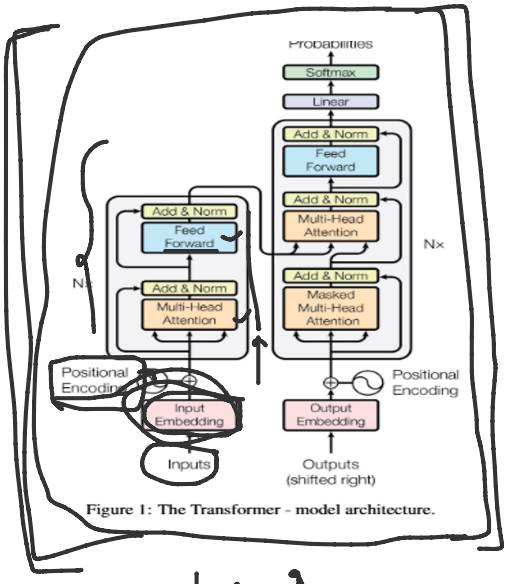


Figure 1: The Transformer - model architecture.

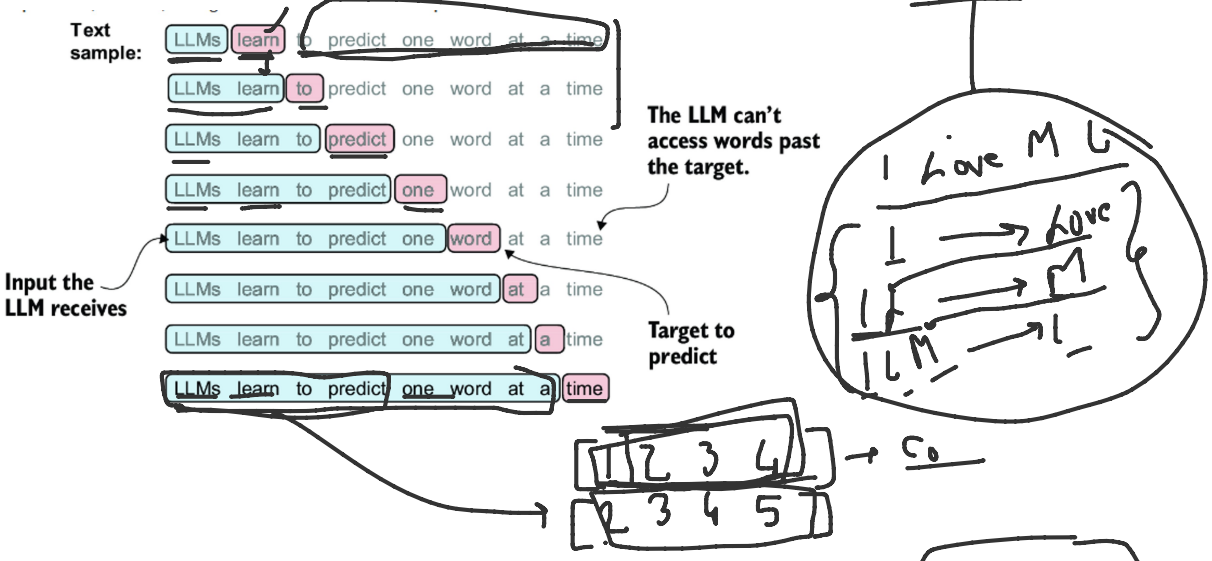
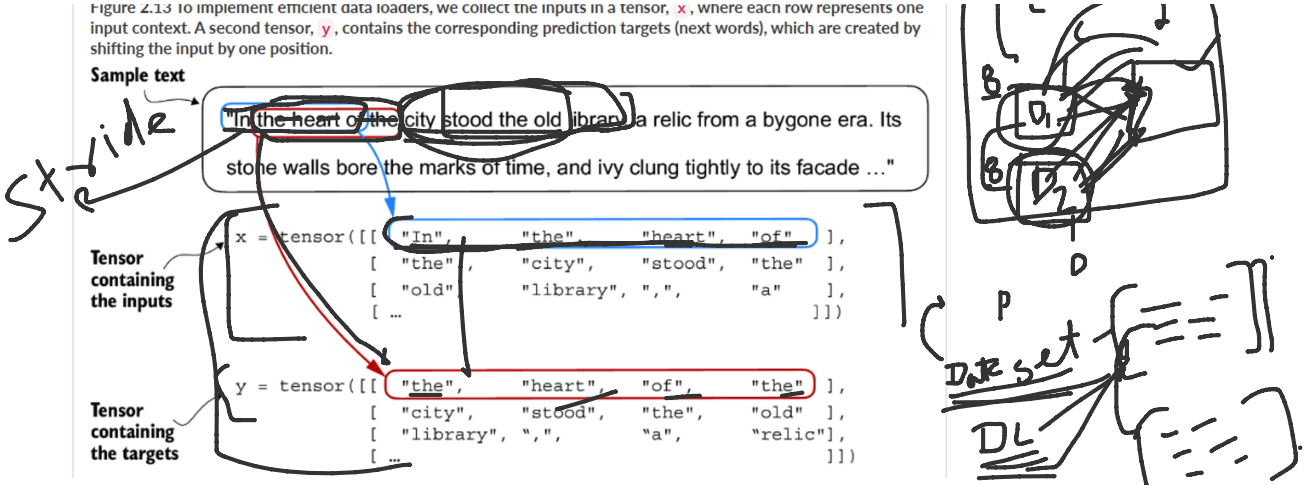
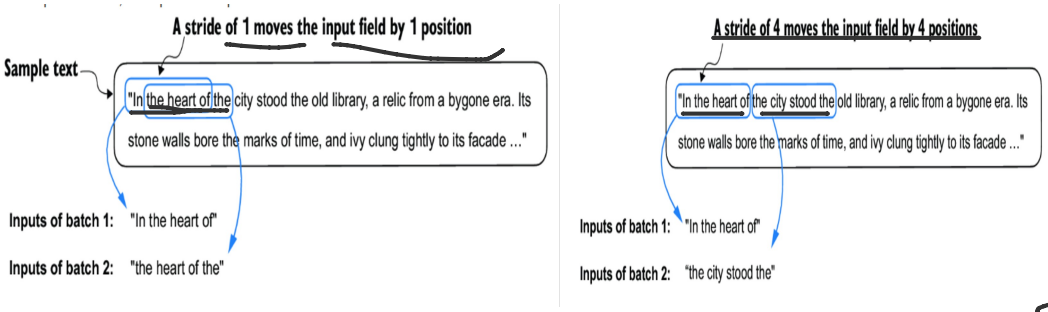


Figure 2.13 To implement efficient data loaders, we collect the inputs in a tensor, `x`, where each row represents one input context. A second tensor, `y`, contains the corresponding prediction targets (next words), which are created by shifting the input by one position.

Sample text

Figure 2.13 To implement efficient data loaders, we collect the inputs in a tensor, `x`, where each row represents one input context. A second tensor, `y`, contains the corresponding prediction targets (next words), which are created by shifting the input by one position.





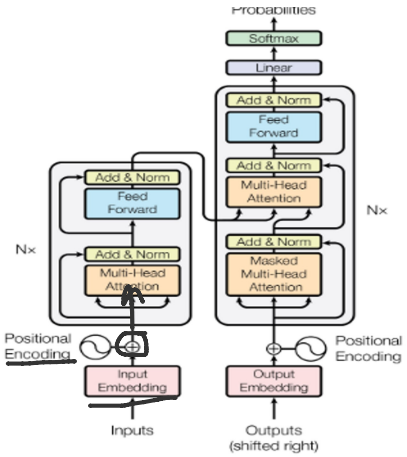
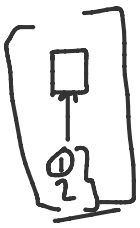
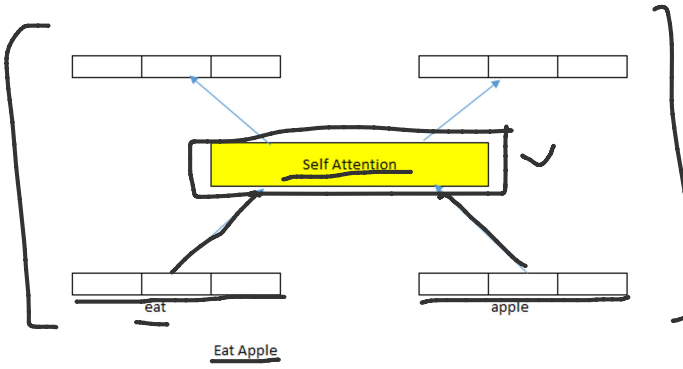


Figure 1: The Transformer - model architecture.

Positional Encoding

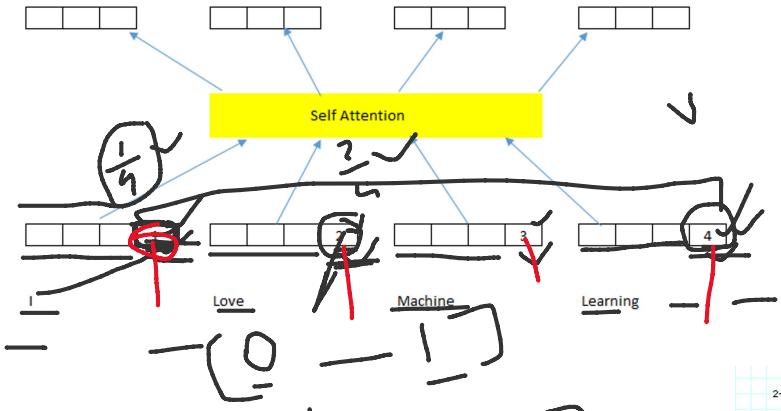




Eat Apple

Issue : Parallel input so we loose the importance of sequence

Lets See a simple solution we can create in order to provide positional information to self attention block

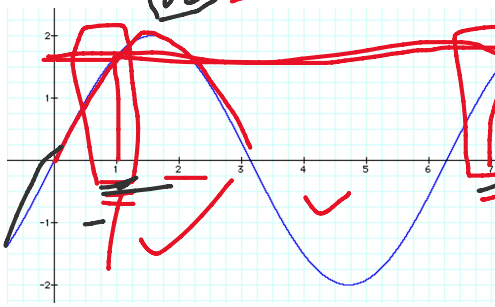


Issue with this approach is positions are unbounded
Means they can go upto any numbers. It can go upto millions

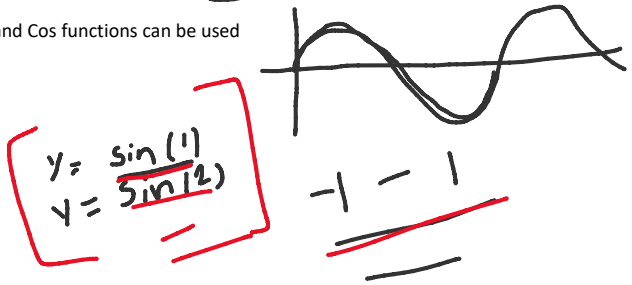
To avoid this we can normalize positional numbers

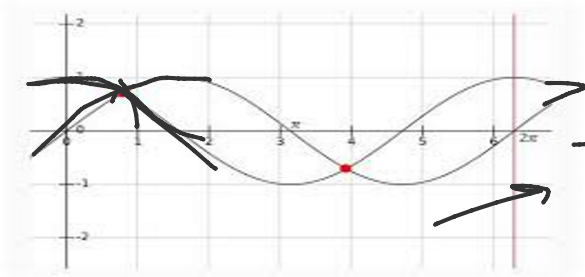
But Input sentence length can be different every time :
Excuse me?

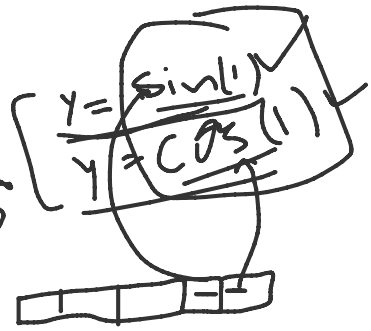
I love Machine Learning

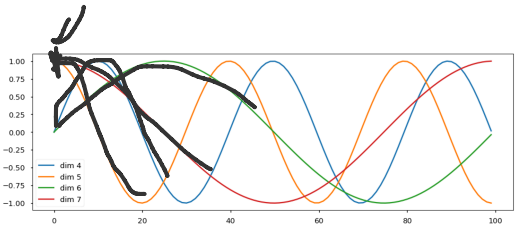


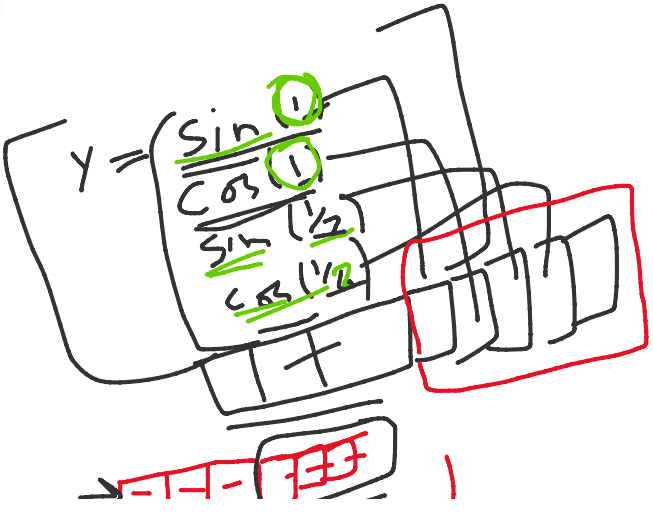
Sin and Cos functions can be used











Dimension of Position Embedding vector = dimension of vector embedding	

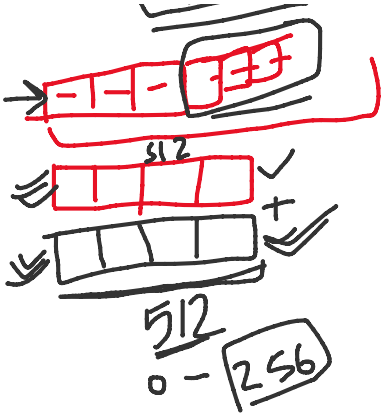
In Attention is all you need paper in stead of concatenating Vector embedding and Position Embedding

Dimension of Position Embedding vector = dimension of vector embedding

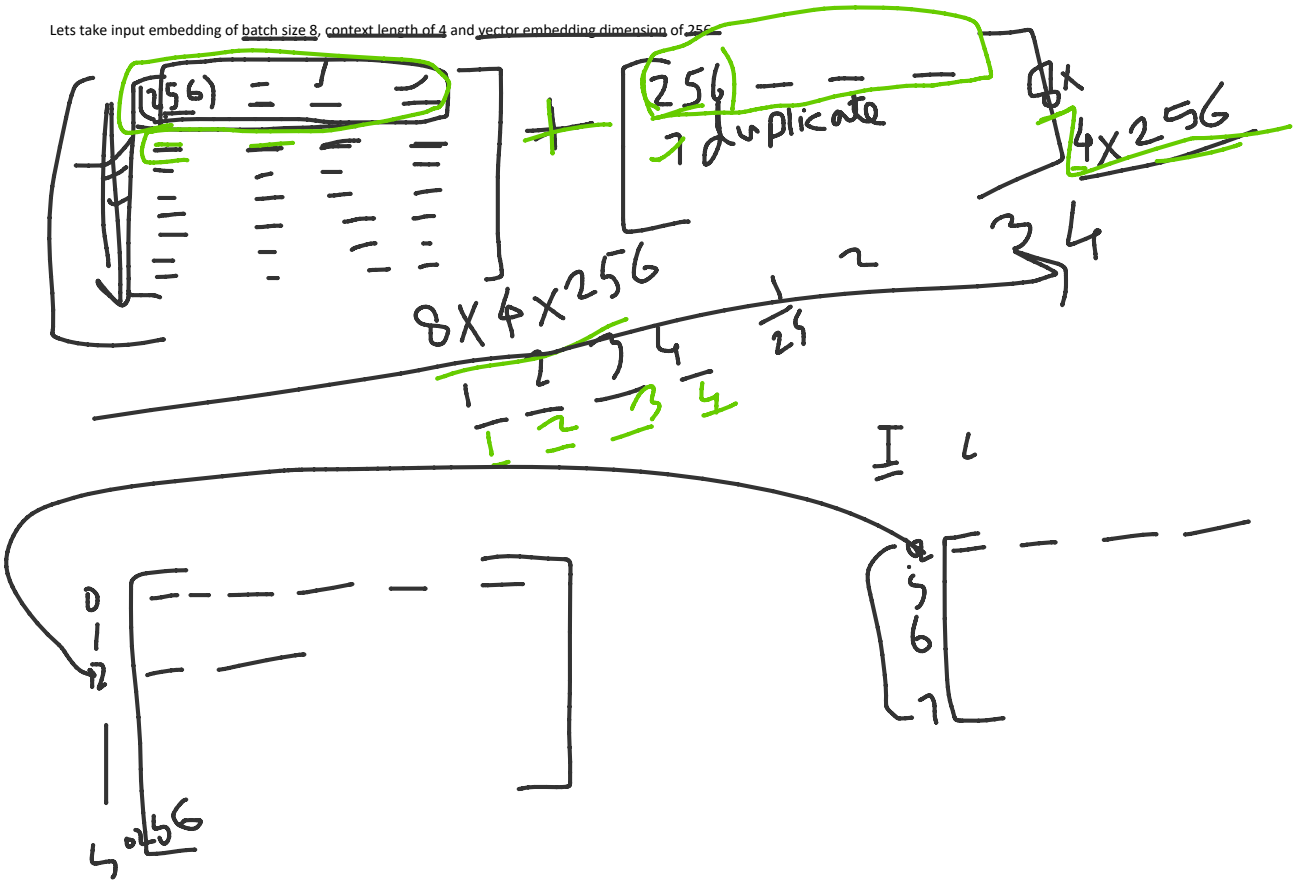
In Attention is all you need paper in stead of concatenating Vector embedding and Position Embedding they are added to avoid increase in dimensions.

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}})$$



Lets take input embedding of batch size 8, context length of 4 and vector embedding dimension of 256



Agenda : Attention, Self Attention and Multi Head Attention : Theory

Recap

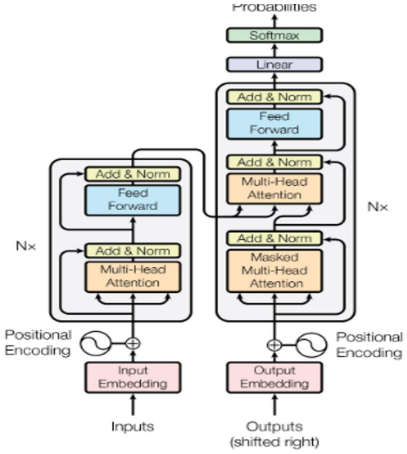
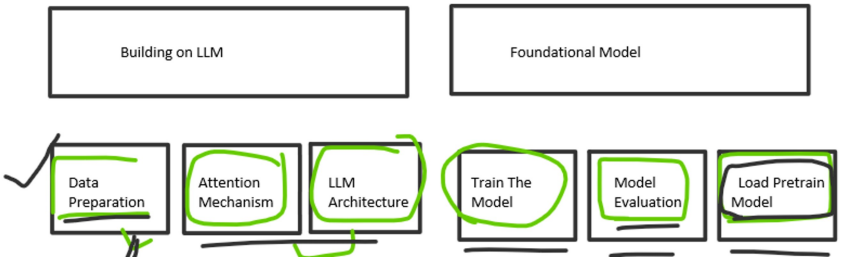
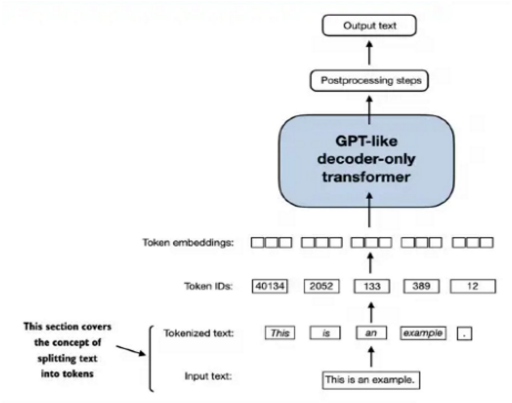
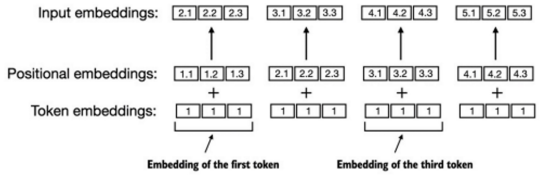


Figure 1: The Transformer - model architecture.







Lets Start Attention Mechanism