

Introduction To Transformers

- 1) RNN / LSTM / GRU RNN
- 2) Encoder Decoder Architecture
- 3) ATTENTION MECHANISM
- 4) TRANSFORMERS

① Why Transformers?

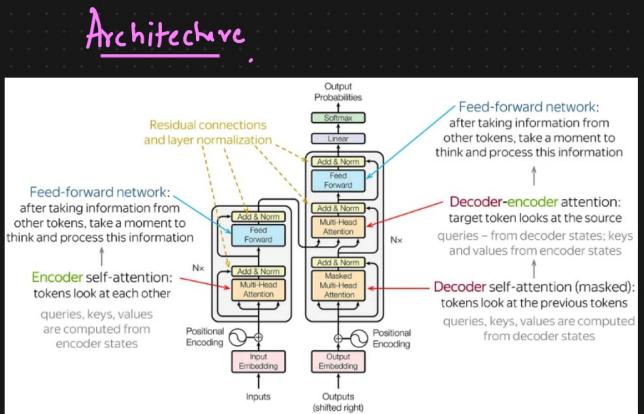
② Architecture of Transformers?

③ SELF ATTENTION $\rightarrow Q, K, V$

④ Positional Encoding

⑤ Multi Head ATTENTION

⑥ Combining the Working of Transformers



Generative AI \rightarrow LM, Multimodel

BERT, GPT \leftarrow

Open AI \rightarrow ChatGPT

GPT-4o

① What And Why \rightarrow Transformers

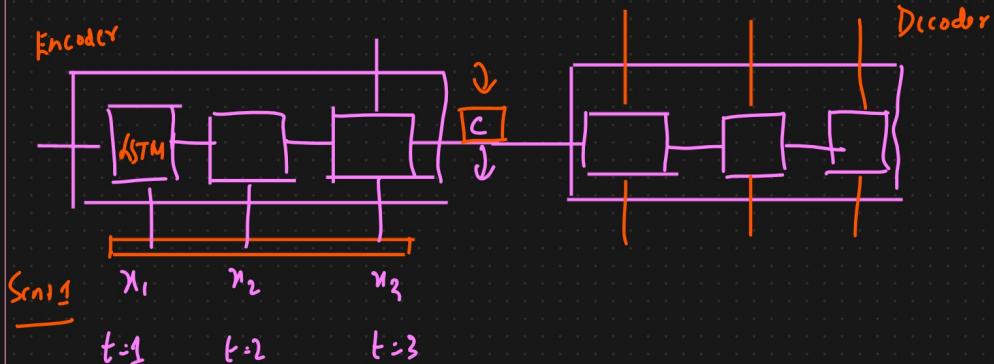
Transformers in natural language processing (NLP) are a type of deep learning model that use self-attention mechanisms to analyze and process natural language data. They are encoder-decoder models that can be used for many applications, including machine translation. \Rightarrow Seq2Seq Task

Eg: Language Translation \rightarrow Google Translation

English \rightarrow French

i/p \Rightarrow Many \rightarrow o/p: Many. $\{$ Length of the sentence $\}$.

Encoder - Decoder



Sentence length $\uparrow\uparrow$

Beam Score $\downarrow\downarrow$

Length Sentence $\uparrow\uparrow$

3.1 DECODER: GENERAL DESCRIPTION

In a new model architecture, we define each conditional probability in Eq. (2) as:

$$p(y_i | y_1, \dots, y_{i-1}, s_i, c_i), \quad (4)$$

where s_i is an RNN hidden state for time i , computed by

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

It should be noted that unlike the existing encoder-decoder approach (see Eq. (2)), here the probability is conditioned on a distinct context vector c_i for each target word y_i .

The context vector c_i depends on a sequence of annotations (h_1, \dots, h_{T_s}) to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i -th word of the input sequence. We explain in detail how the annotations are computed in the next section.

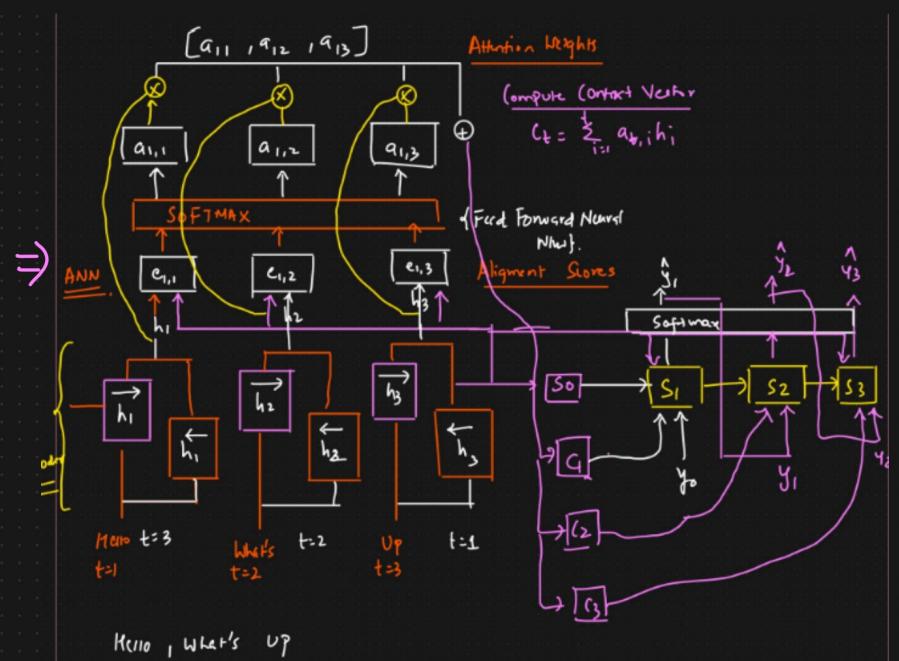
The context vector c_i is, then, computed as a weighted sum of these annotations h_i :

$$c_i = \sum_{j=1}^{T_s} \alpha_{ij} h_j. \quad (5)$$

Figure 1: The graphical illustration of the proposed model trying to generate the i -th target word y_i given a source sentence (x_1, x_2, \dots, x_T) .

Additional Context \rightarrow Decoder

long Sentence Accuracy ↑↑



Attention Mechanism

① Parallelly we cannot send all the words in a sentence \rightarrow Scalable

DATASET \rightarrow Huge \rightarrow Scalable With Respect to Training.

TRANSFORMERS \neq LSTM RNN

Self Attention Module \leftarrow All the words will be parallelly sent to encoder.



Positional Encoding

Transformer ↑ DATASET \rightarrow Amazing SOTA \leftarrow NLP

Transfer Learning \rightarrow MultiModal Task \rightarrow NLP + Image \leftarrow

Transformers \div AI Space \rightarrow SOTA Model \rightarrow



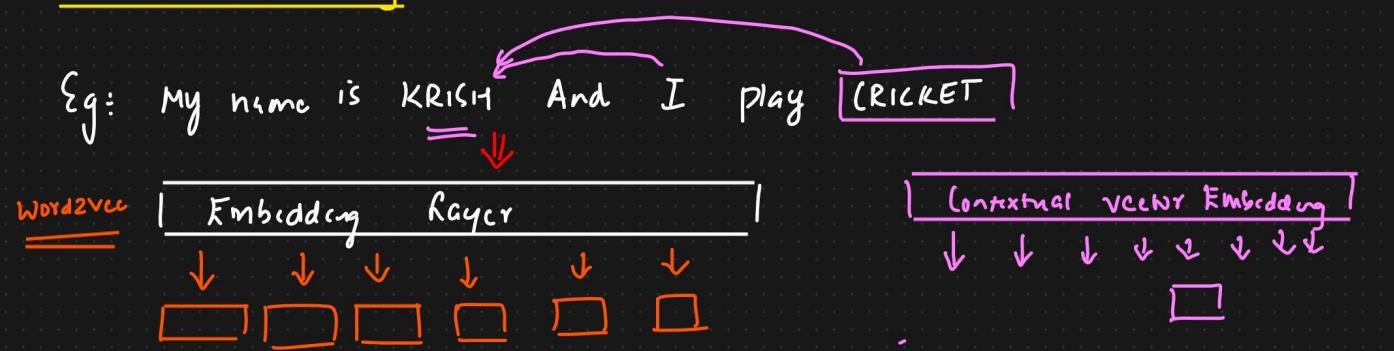
BERT GPT \rightarrow Transfer learning \rightarrow SOTA Models \rightarrow DALLE \leftarrow Generating AI

Train huge Data

AI's

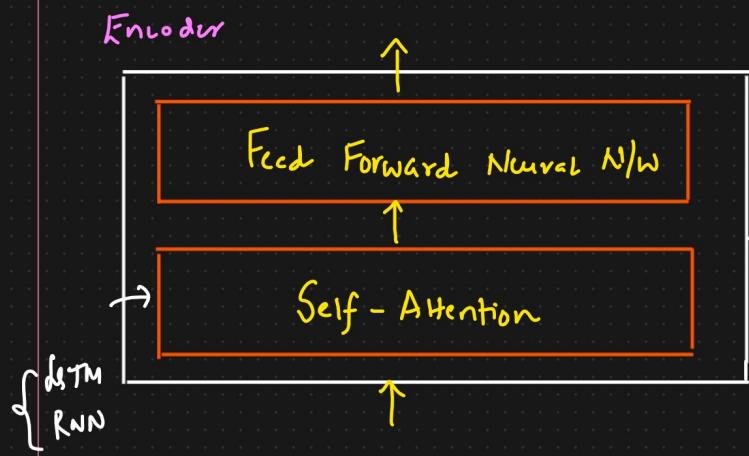
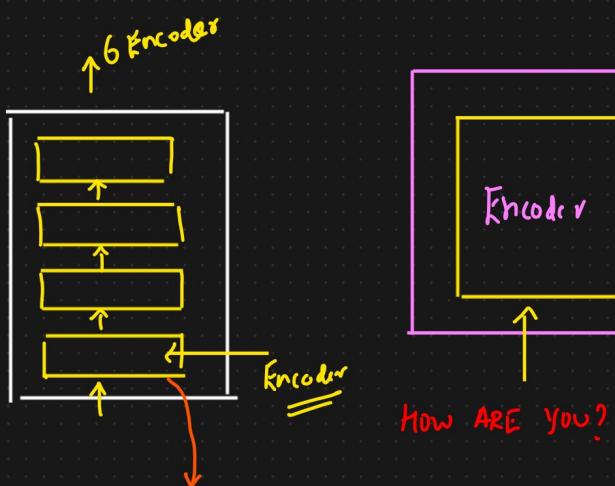
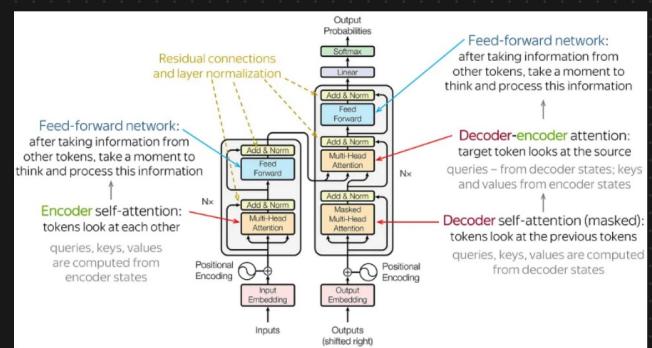
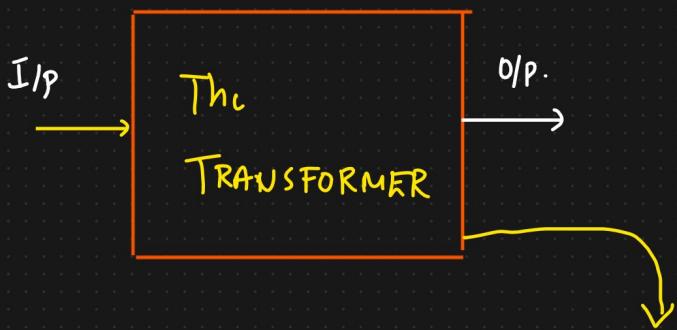
Generating AI

② Contextual Embedding → Self Attention

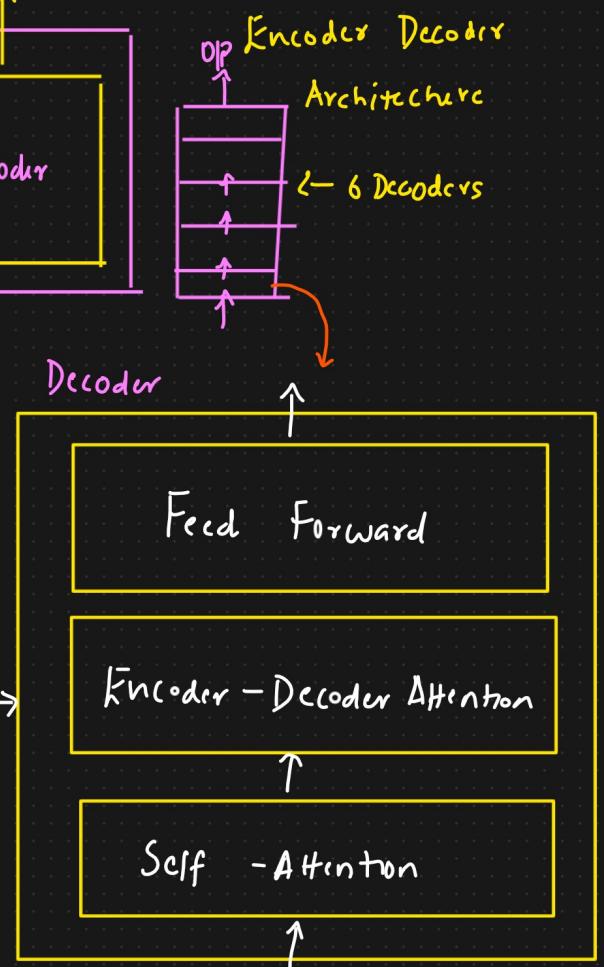


② Basic Transformer Architecture

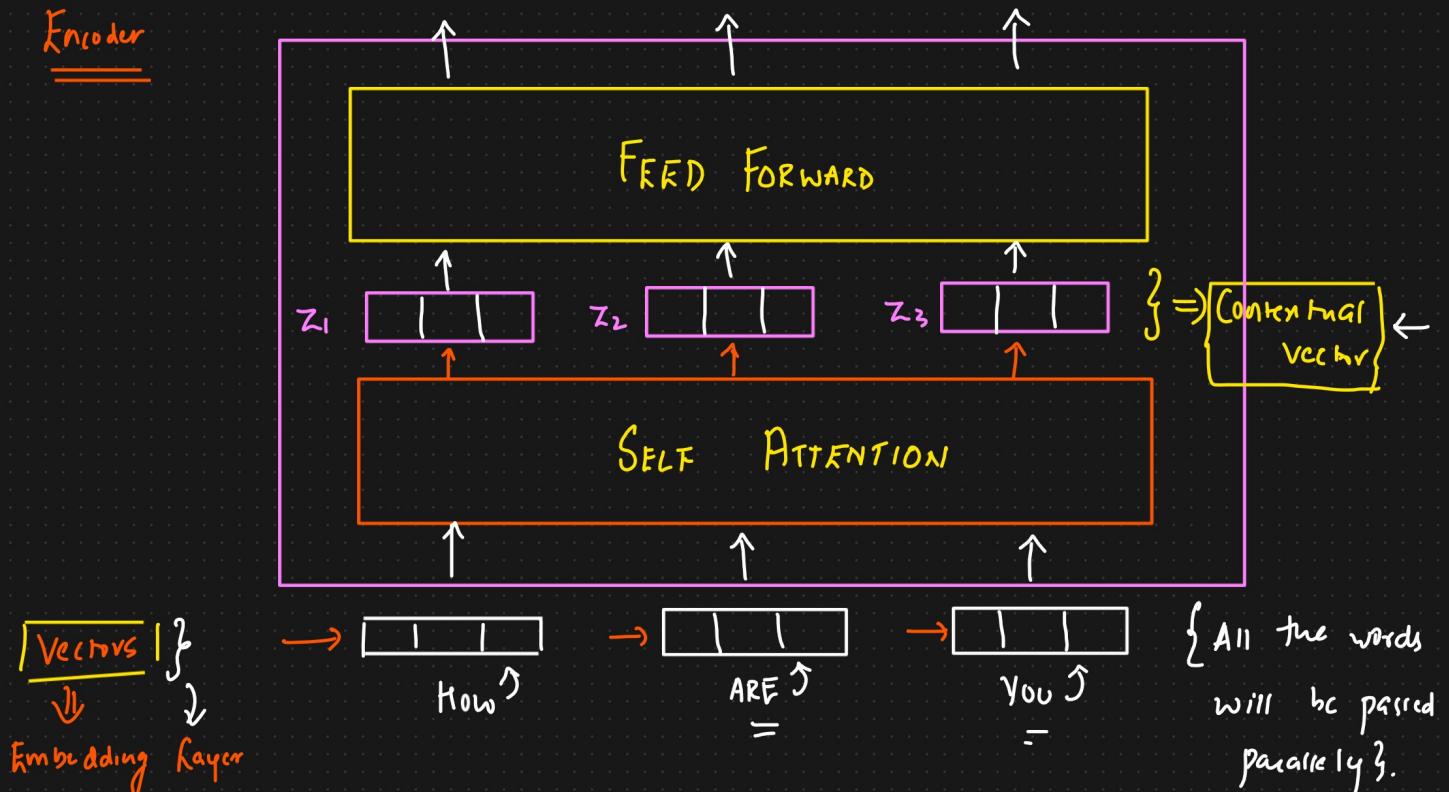
{Seq2Seq Task} → Language Translation {Eng → French}



Comment vas-tu?

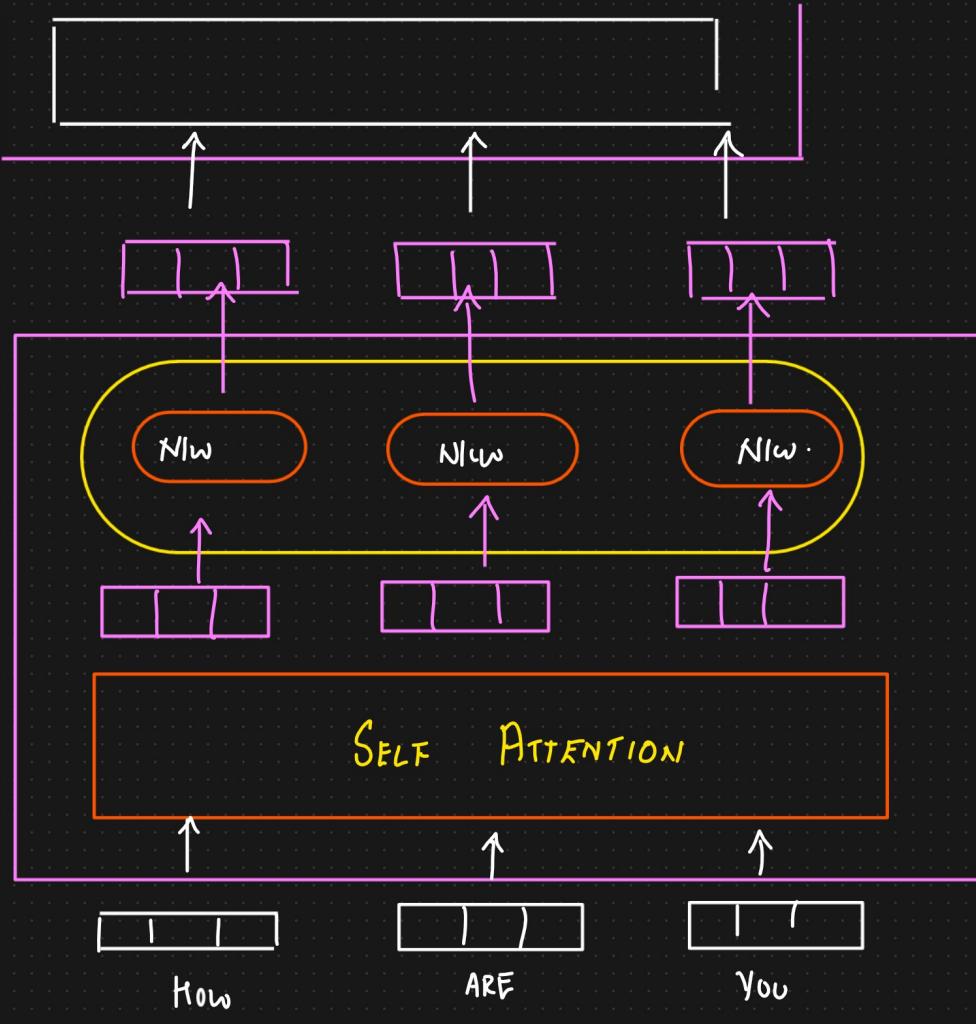


Encoder



Encoder¹

Encoder



Self Attention At a Higher Level

Eg.: The cat sat on the mat, the cat lay on the rug.

Word Embedding
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓

SELF ATTENTION

↓
The.

The

→ [Cat] → [Cat] → [| | |]
| → [Sat] Sat

{ Contextual Embedding }

Rank 1 → On
= 2 → the
3 → the
mat

mat → [- | - | -]



Self Attention In Detail

① To Create 3 vectors from each of the encoder i/p. Query vector,

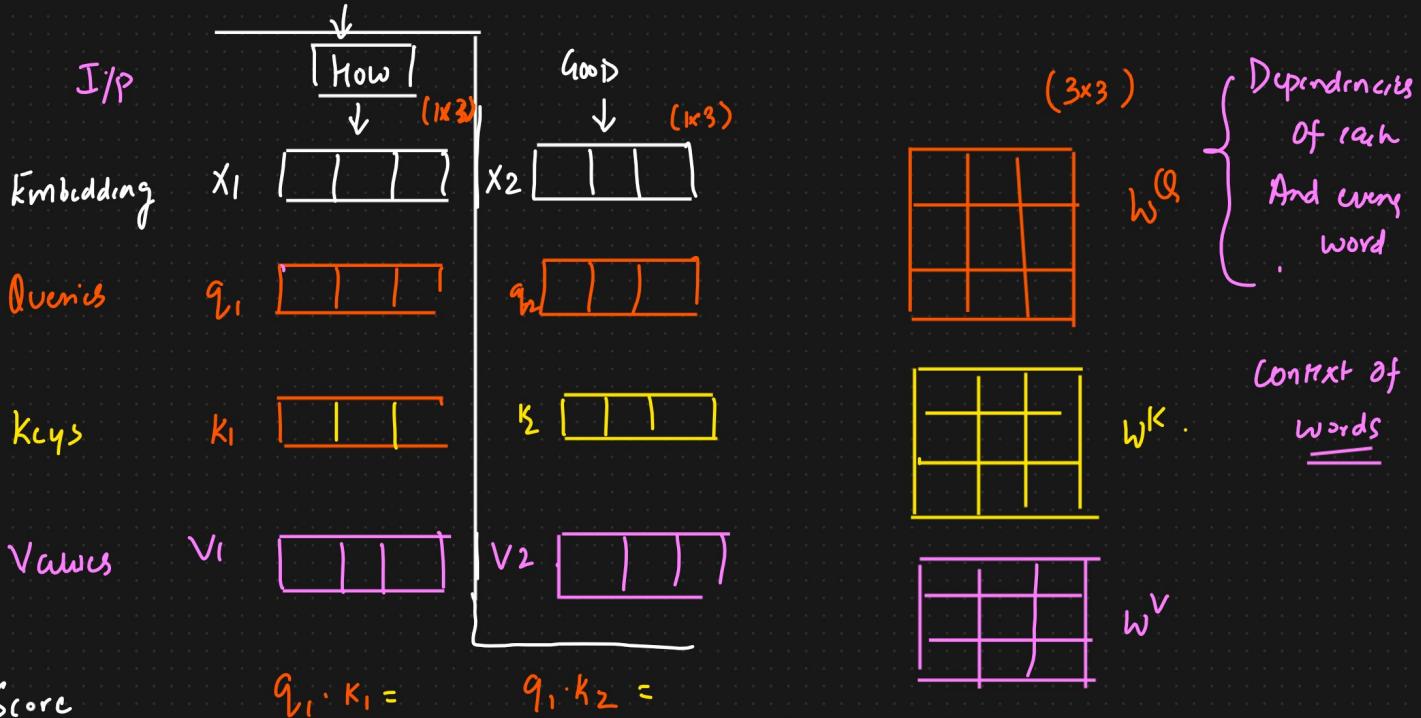
Key vector, value vector. \Rightarrow Contextual Embedding

Eg.: YT \Rightarrow Search keywords \Rightarrow {Query}.

Query \rightarrow {Key} \rightarrow Tags
 \rightarrow Description \Rightarrow Value \Rightarrow o/p Video

② Second step in calculating self attention is to calculate the Score.

The Score determines how much focus to place on the other part of the sentence.



$$\text{Score} \quad q_1 \cdot k_1 = \quad q_1 \cdot k_2 =$$

④ How much focus to place on other part of the I/P Sentence as we encode a word at certain position

Divide \sqrt{dk}

More stable gradients

SoftMax

Dimension = 3 Dimension $\uparrow = T/2$

\downarrow

$\frac{\text{Value}}{\sqrt{dk}}$ \approx $\frac{\text{Value}}{\sqrt{dk}}$ \Rightarrow

$\frac{[0-1]}{\sqrt{dk}}$ $\frac{[0-1]}{\sqrt{dk}}$

$0.88 + 0.12 = 1$

Softmax \Rightarrow The Softmax score determines how much each word will be expressed at this position

X X X

Value Vectors v_1 [| |] v_2 [| | |]

0.88 [| | |]

$\frac{0.88}{0.88 + 0.12}$ + $\frac{0.12}{0.88 + 0.12}$

\downarrow \downarrow

Contextual Vector $\xrightarrow{T_1}$ [| | |] $\xrightarrow{T_2}$ [| | |]

Self Attention At Higher And Detailed Level

Self-attention, also known as scaled dot-product attention, is a crucial mechanism in the transformer architecture that allows the model to weigh the importance of different tokens in the input sequence relative to each other.

Idea :

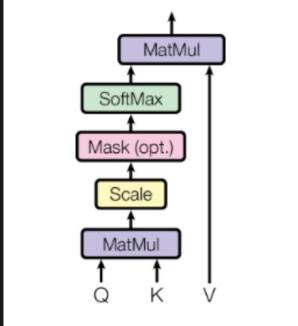


Language Translation Self Attention

Text Summarization Sentence, Dataset

Embedding Layer → Fixed Vector

Scaled Dot-Product Attention



i) Inputs: Queries, Keys, And Values

Model → Queries, Keys And Values

1. Query Vectors (Q):

Role: Query vectors represent the token for which we are calculating the attention. They help determine the importance of other tokens in the context of the current token.

Importance:

Focus Determination: Queries help the model decide which parts of the sequence to focus on for each specific token. By calculating the dot product between a query vector and all key vectors, the model assesses how much attention to give to each token relative to the current token.

Contextual Understanding: Queries contribute to understanding the relationship between the current token and the rest of the sequence, which is essential for capturing dependencies and context.

2. Key Vectors (K):

Role: Key vectors represent all the tokens in the sequence and are used to compare with the query vectors to calculate attention scores.

Importance:

Relevance Measurement: Keys are compared with queries to measure the relevance or compatibility of each token with the current token. This comparison helps in determining how much attention each token should receive.

Information Retrieval: Keys play a critical role in retrieving the most relevant information from the sequence by providing a basis for the attention mechanism to compute similarity scores.

3. Value Vectors (V):

Role: Value vectors hold the actual information that will be aggregated to form the output of the attention mechanism.

Importance:

Information Aggregation: Values contain the data that will be weighted by the attention scores. The weighted sum of values forms the output of the self-attention mechanism, which is then passed on to the next layers in the network.

Context Preservation: By weighting the values according to the attention scores, the model preserves and aggregates relevant context from the entire sequence, which is crucial for tasks like translation, summarization, and more.

$$\text{Input Sequence} = \left[\text{"The"}, \text{"CAT"}, \text{"SAT"} \right]$$

$$\text{Embedding Size} = 4$$

$$Q, K, V \Rightarrow 4$$

$$\xrightarrow{\text{L}} \boxed{\text{SIFT AII}} \rightarrow \boxed{\text{OPIO}}$$



$$\boxed{\text{IPO}} \quad \boxed{\square} \quad \boxed{\square}$$

↓
Sentence, Dataset

① Token Embedding

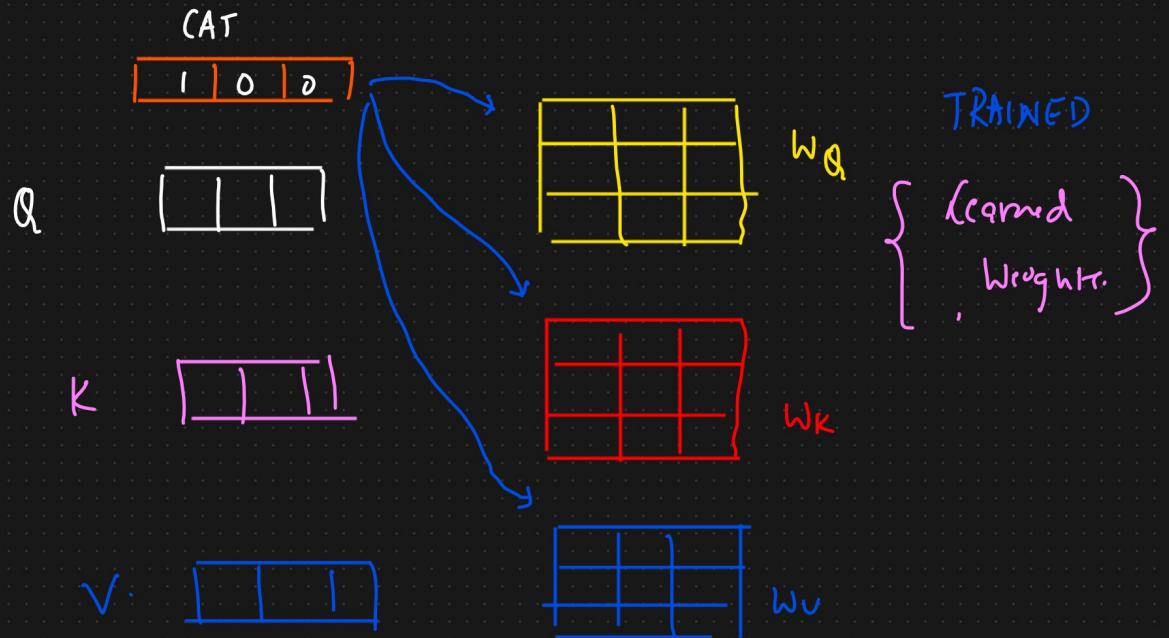
$$E_{\text{The}} = [1 \ 0 \ 1 \ 0]$$

$$E_{\text{SAT}} = [1, 1, 1, 1]$$

$$E_{\text{CAT}} = [0 \ 1 \ 0 \ 1]$$

② Linear Transformation

We create Q, K, V by multiplying the embeddings by learned weights matrices W_Q , W_K and W_V .



Let's consider

$$W_Q = W_K = W_V = I \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \leftarrow$$

$$Q_{\text{The}} = [1 \ 0 \ 1 \ 0] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1 \ 0 \ 1 \ 0]$$

$$K_{\text{The}} = [1 \ 0 \ 1 \ 0]$$

$$V_{\text{The}} = [1 \ 0 \ 1 \ 0]$$

$$\textcircled{1} \quad Q_{\text{The}} = K_{\text{The}} = V_{\text{The}} = [1 \ 0 \ 1 \ 0]$$

$$\textcircled{2} \quad Q_{\text{CAT}} = K_{\text{CAT}} = V_{\text{CAT}} = [0 \ 1 \ 0 \ 1]$$

$$\textcircled{3} \quad Q_{\text{SAT}} = K_{\text{SAT}} = V_{\text{SAT}} = [1, 1, 1, 1]$$

③ Compute Attention Scores

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

The

$$\text{Score}(Q_{\text{The}}, K_{\text{The}}) = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T = 2$$

$$\text{Score}(Q_{\text{The}}, K_{\text{CAT}}) = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T = 0$$

$$\text{Score}(Q_{\text{The}}, K_{\text{SAT}}) = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T = 2$$

For the token CAT

$$\text{Score}(Q_{\text{CAT}}, K_{\text{The}}) = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T = 0$$

$$\text{Score}(Q_{\text{CAT}}, K_{\text{CAT}}) = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}^T = 2$$

$$\text{Score}(Q_{\text{CAT}}, K_{\text{SAT}}) = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T = 2.$$

For the Token SAT

$$\left\{ \begin{array}{l} \text{Score}(Q_{\text{SAT}}, K_{\text{The}}) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix}^T = 2 \\ \text{Score}(Q_{\text{SAT}}, K_{\text{CAT}}) = 2 \\ \text{Score}(Q_{\text{SAT}}, K_{\text{SAT}}) = 4 \end{array} \right.$$

④ Scaling

We take up the scores and scale down by dividing the scores by the

$$\sqrt{d_K} \Rightarrow d_K = 4 \quad \sqrt{d_K} = 2.$$

Scaling in the attention mechanism is crucial to prevent the dot product from growing too large. \Rightarrow Ensure stable gradients during Training.

d_K is large \rightarrow

① Gradient Exploding

② Softmax Saturation $\{\curvearrowright\}$ \rightarrow Vanishing Gradient Problem.

$$Q = \begin{bmatrix} 2 & 3 & 4 & 1 \end{bmatrix} \quad K_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \quad K_2 = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix}$$

Without Scaling

$$Q \cdot K_1^T = 2 \times 1 + 3 \times 0 + 4 \times 1 + 1 \times 0 = 2 + 4 = 6.$$

$$Q \cdot K_2^T = 2 \times 0 + 3 \times 1 + 4 \times 0 + 1 \times 1 = 0 + 3 + 0 + 1 = 4$$

* Score $[6, 4] \Rightarrow$ Scaling Not Applied

$$\text{Softmax}([6, 4]) = \left[\frac{e^6}{e^6 + e^4}, \frac{e^4}{e^6 + e^4} \right] = \left[\frac{e^6}{e^6(1 + e^{-2})}, \frac{e^4}{e^4(e^2 + 1)} \right]$$

① Property of Softmax w_Q, w_K, w_V

$$([10, 1]) = \left[\frac{0.99}{\cancel{10}}, \frac{\cancel{0.01}}{\cancel{1}} \right] = \left[\frac{1}{(1 + e^{-2})}, \frac{1}{(e^2 + 1)} \right]$$

Dot product = Large values $\approx [0.88, 0.12]$.

Most of the attention weight is assigned to the first key vector,
very little to the second vector,

With Scaling

① Compute Scaled Dot Product

$$[6, 4] \Rightarrow \text{Scale} \Rightarrow \left[\frac{6}{\sqrt{2}}, \frac{4}{\sqrt{2}} \right] = \left[3, 2 \right]. \quad \frac{\sqrt{d_K}}{\sqrt{d_K}} \uparrow \stackrel{\text{dimension} \uparrow \text{variance} \uparrow}{\text{same}}$$

$$\text{Softmax}([3, 2]) = \left[\frac{e^3}{e^3 + e^2}, \frac{e^2}{e^3 + e^2} \right] = \left[\frac{e^3}{e^3(1 + e^{-1})}, \frac{e^2}{e^2(e^1 + 1)} \right] = [0.73, 0.27] \Rightarrow \text{Attention weights}$$

(4) Here, the attention weights are more balanced compared to the unscaled case

Summary of Importance

Stabilizing Training: Scaling prevents extremely large dot products, which helps in stabilizing the gradients during backpropagation, making the training process more stable and efficient.

Preventing Saturation: By scaling the dot products, the softmax function produces more balanced attention weights, preventing the model from focusing too heavily on a single token and ignoring others.

Improved Learning: Balanced attention weights enable the model to learn better representations by considering multiple relevant tokens in the sequence, leading to better performance on tasks that require context understanding.

Scaling ensures that the dot products are kept within a range that allows the softmax function to operate effectively, providing a more balanced distribution of attention weights and improving the overall learning process of the model.

$$(4) \text{Scaling} = \sqrt{d_K} = \sqrt{4} \Rightarrow 2$$

Similarly Scaling

$$\text{Scaled-Score } (Q_{\text{The}}, K_{\text{The}}) = 2/2 = 1$$

will be done for
all other tokens.

$$\text{Scaled-Score } (Q_{\text{The}}, K_{\text{CAT}}) = 0/2 = 0$$

$$\text{Scaled-Score } (Q_{\text{The}}, K_{\text{SAT}}) = 2/2 = 1$$

(5) Apply Softmax

$$\text{ATTENTION WEIGHTS}_{\text{"The"}} = \text{Softmax}([1, 0, 1]) = [0.4223, 0.1554, 0.4223]$$

$$\text{ATTENTION WEIGHTS}_{\text{"CAT"}} = \text{Softmax}([0, 2, 2]) = [0.1554, 0.4223, 0.4223]$$

$$\text{ATTENTION WEIGHTS}_{\text{"SAT"}} = \text{Softmax}([2, 2, 4]) = [0.2119, 0.2119, 0.5762]$$

(6) Weight Sum of Values

We multiply the attention weights by corresponding value vectors

For the Token The =

$$\text{Output}_{(\text{The})} = 0.4223 * V_{\text{The}} + 0.1554 * V_{\text{CAT}} + 0.4223 * V_{\text{Sal.}}$$

$$= 0.4223 [1 \ 0 \ 1 0] + 0.1554 [0 \ 1 0 1] + 0.4223 [1 1 1]$$

$$= [0.4223, 0, 0.4223, 0] + [0, 0.1554, 0, 0.1554] + [0.4223, 0.4223, \\ 0.4223, 0.4223]$$

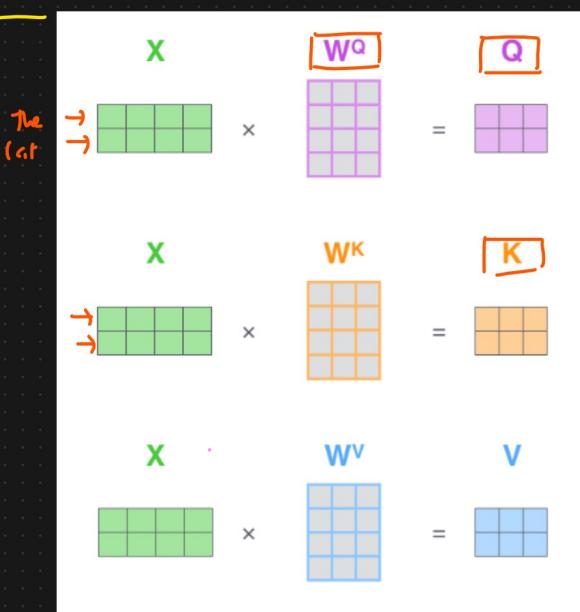
$$= [1.2669, 0.9999, 1.2669, 0.9999].$$

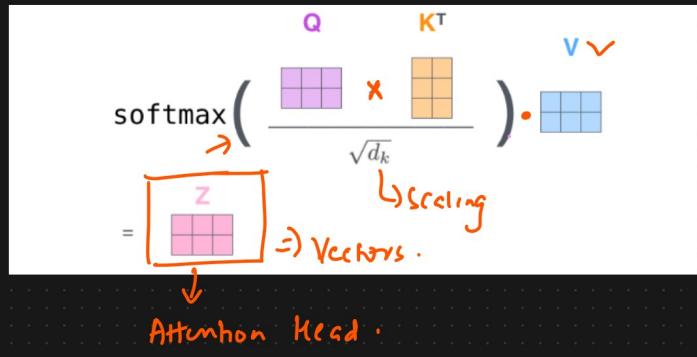
↓ Contextual
vector

The 1 1 0 1 1 0 \Rightarrow Self Attention $\Rightarrow [1.2669, 0.9999, 1.2669, 0.9999].$

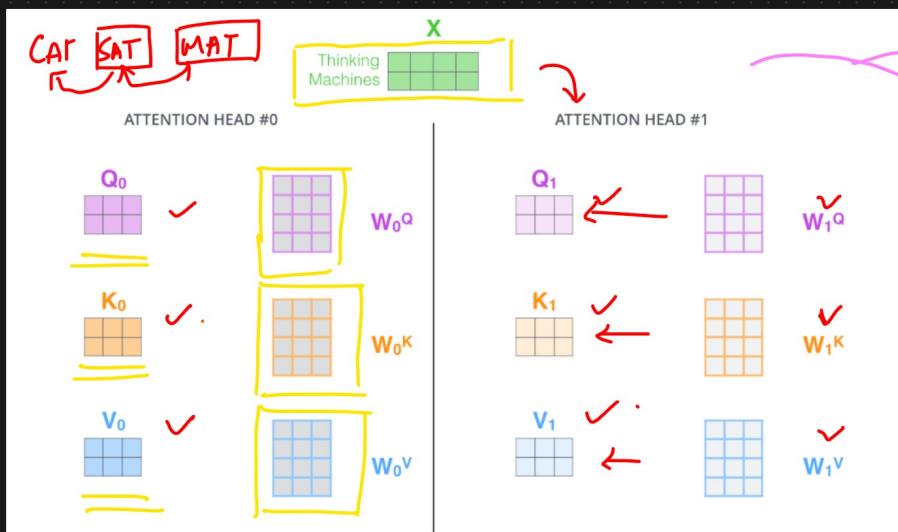
- ① $\hookrightarrow Q, K, V [w^Q, w^K, w^V]$
② \hookrightarrow Attention Score
③ \hookrightarrow Scaled
④ \hookrightarrow Softmax
⑤ \hookrightarrow Weighted Sum of Value (Softmax \times V)

④ Multi Head Attention



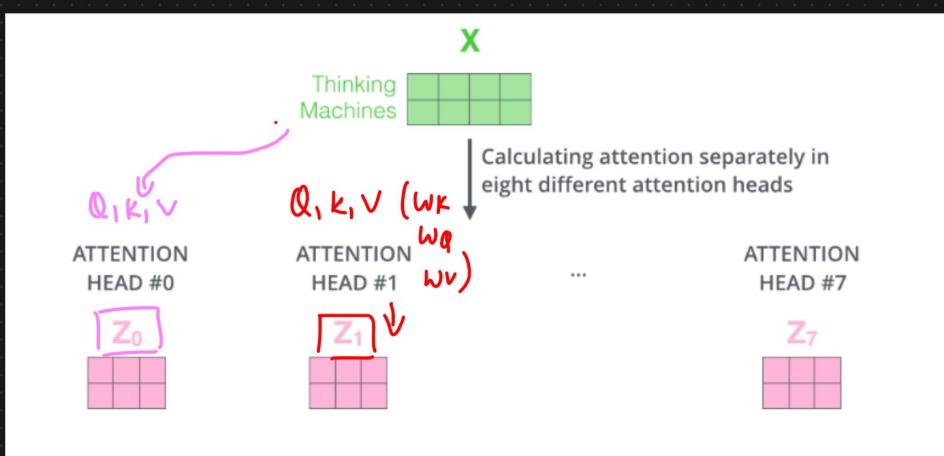


→ Self Attention with Multi Heads



Score, Softmax $\times V$
 \downarrow \downarrow
 $[\dots]$ $[\dots]$
 Z_0 Z_1
 Vectors

Multi Head Attention



$[z]$

$[z]$

$[z]$

⑥ Feed Forward Neural Network

