Attention is All You Need

What and why to use Transformers?

Transformers are a type of model that uses a self-attention mechanism to analyze and process natural language data. They are encoder-decoder models that can be used for many applications, including machine translation.

when we are using the encoder and decoder, we can't pass the entire sentence parallelly & in attention mechanism we can pass the entire sentence parallelly.

The groundbreaking paper **"Attention Is All You Need"** was written by a team of researchers from Google Brain, including **Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser,** and **Illia Polosukhin**. Published in 2017, this paper introduced the **Transformer model**, a revolutionary architecture that replaced the need for recurrent and convolutional layers in sequence-to-sequence tasks.

1. **Self-Attention**
   A mechanism where each token in the input sequence attends to all other tokens in the sequence to compute its representation. Self-attention allows the model to weigh the importance of each word in a sequence relative to others.
2. **Scaled Dot-Product Attention**
   The specific implementation of the attention mechanism used in Transformers. It computes the attention scores using the dot product of the query and key vectors, scaled by the square root of the dimension of the key vectors.
3. **Multi-Head Attention**
   Instead of having a single attention mechanism, the model employs multiple attention heads, each learning different aspects of the relationship between words in the sequence.
4. **Positional Encoding**
   Since the Transformer model does not have any inherent notion of the order of the tokens (unlike RNNs or LSTMs), positional encodings are added to the input embeddings to provide information about the position of each token in the sequence.
5. **Encoder-Decoder Architecture**
   The Transformer uses an encoder-decoder structure:
   Encoder: Processes the input sequence and produces a sequence of context-rich representations (hidden states).
   Decoder: Takes the encoder output and generates the final output sequence.

6. Masking

A technique used during training to prevent certain tokens from attending to future tokens. For example, in a language model, a token at position t should not be able to attend to tokens at positions

7. Feed-Forward Neural Networks (FFNs)

After the attention mechanism, each token is passed through a feed-forward neural network, which consists of two layers with a ReLU activation in between.

8. Layer Normalization

A technique used to normalize the inputs to each layer, ensuring that the mean and variance of the inputs are kept consistent.

10. Residual Connections: Connections that allow the input to be added to the output of each layer before passing it through the next layer.

11. Position-wise Feed-Forward Networks (FFNs)

A fully connected feed-forward network applied to each position independently. It enables the model to apply the same transformation to each position of the sequence.

13. Attention Score

The value computed during the attention mechanism, indicating how much focus each token in the sequence should have on other tokens. The attention score is used to weight the contribution of each token in the sequence when generating the output.

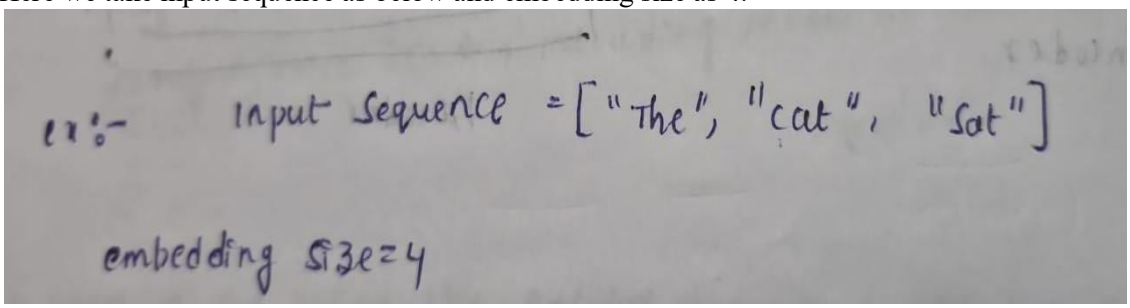
Let us explain with an example:

Encoding Part :

Converting the data into the context vector

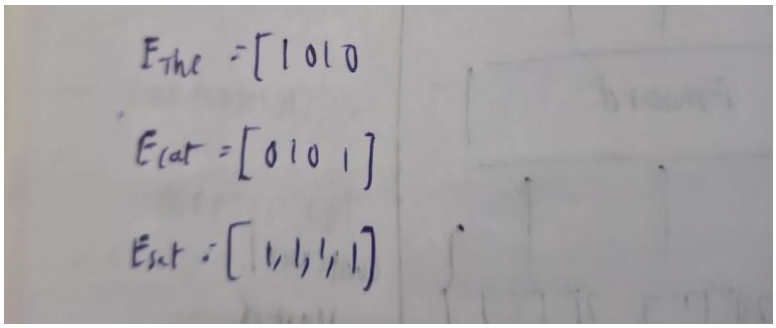
1) inputs

Here we take input sequence as below and embedding size as 4.



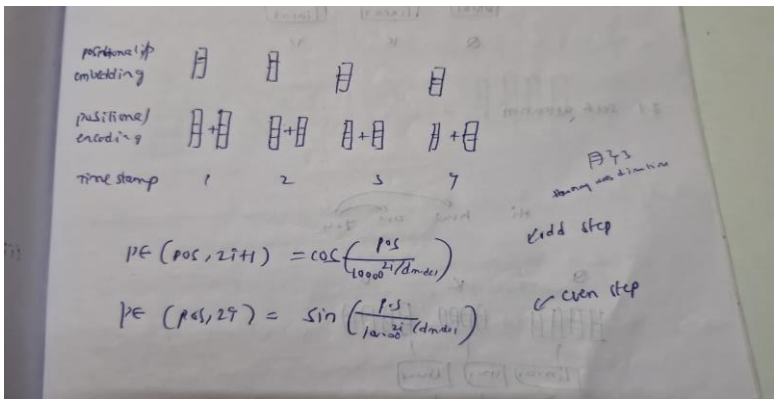
1) Token embeddings

here I used OHE, we can use any type of encoding from hugging face, for our understanding I used OHE.
OHE is used to convert the data into categorical data into numerical data.



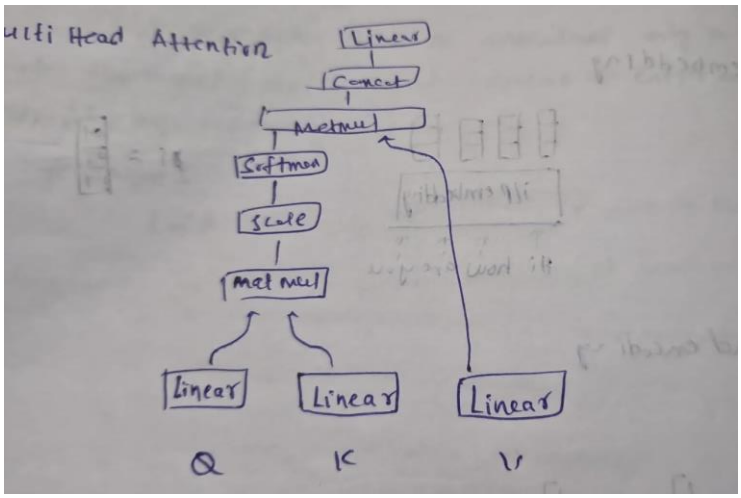
2) Positional Encoding

This are used to attain the context of the sequence, in this task its important to maintain the position also, suppose take an example as translation where translated words are depended on the input words.



3) Multiheaded Attention

Self-attentions with multi-heads we call as multi-head attention. Here we concatenate the all-individual self-attention layers.

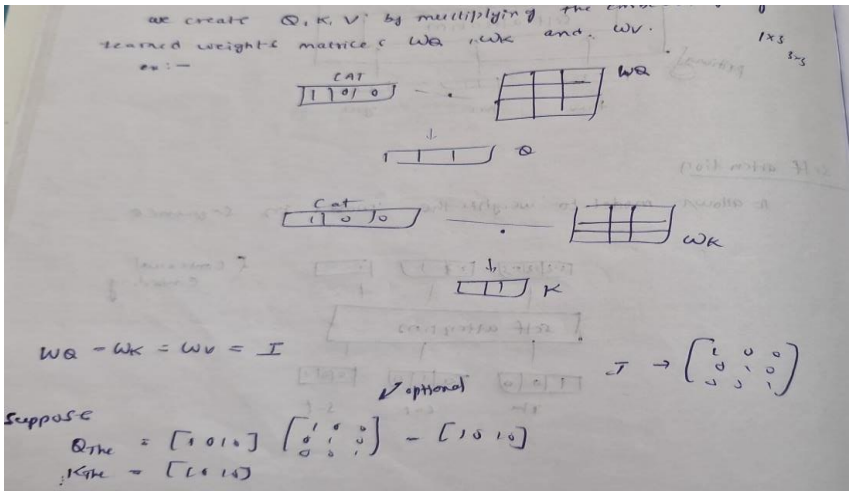


4.1) Q, K, V Matrices

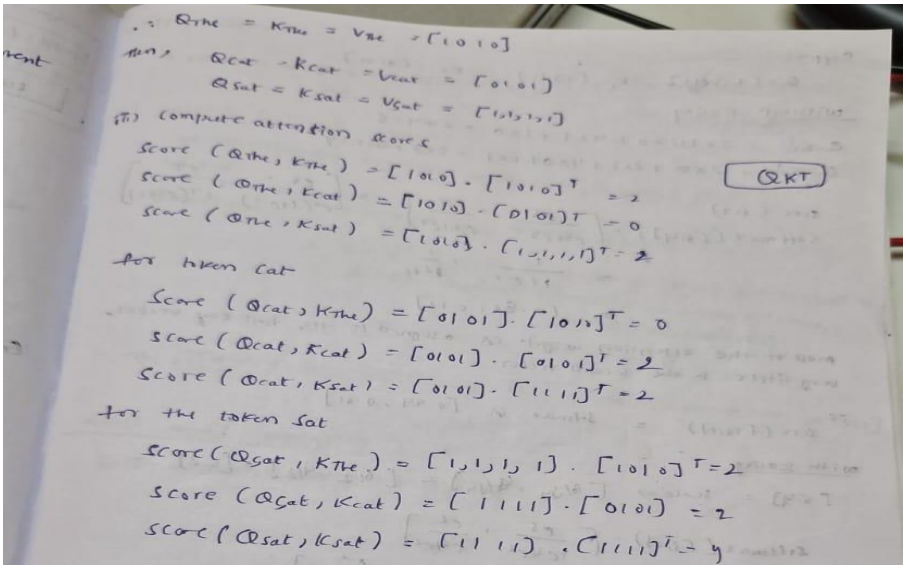
Query (Q): Represents the token for which we are computing the attention.

Key (K): Represents the tokens that we are attending to.

Value (V): Represents the actual information used to generate the output.



4.2) Computing attention scores : it will tell how much of attention you have to give to words in for future purpose.



4.3) Scaling:

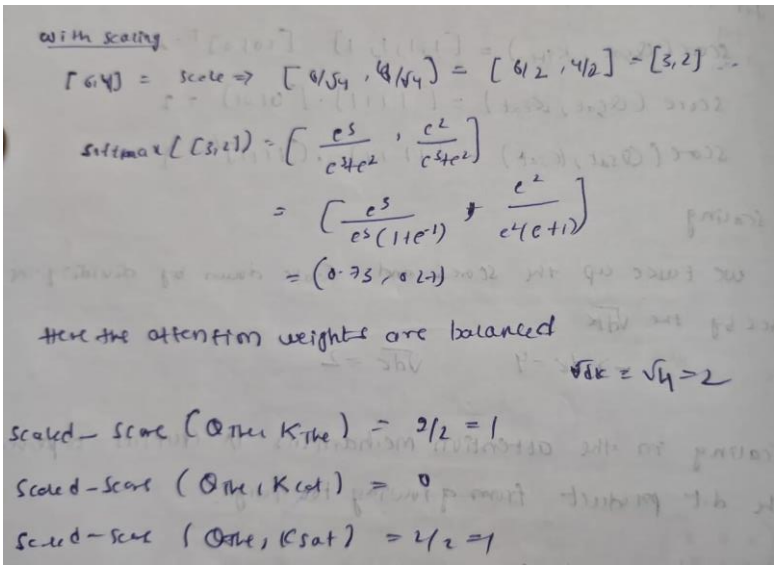
We take up the scores and scale down by dividing the scores by √dk

In our example, we take embedding length as 4

⇨ dk = 4, √dk=√4=2

Scaling in the attention mechanism is crucial to prevent the dot product from growing too large.

If dk is large the it leads to Exploding gradient descent



4.3) Applying SoftMax

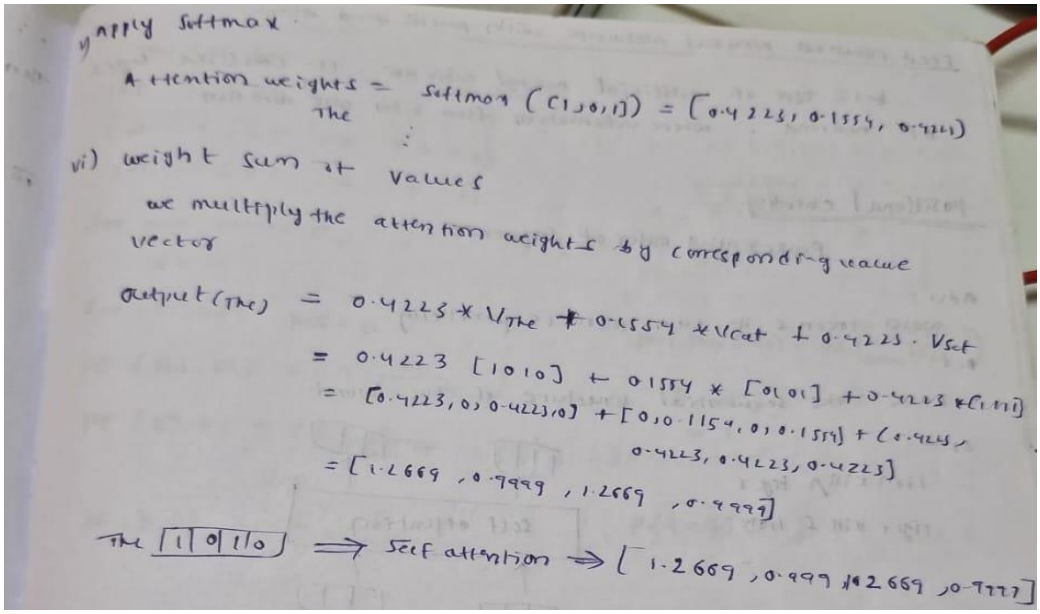
Applying SoftMax is because we have to include the non-linearity to the model, through the it can explore more and give accurate result.

Attentions Weights= SoftMax ([1,0,1]) = [0.4223,0.1554,0.4221]

Like this we do it for others values also

4.4) Weighted sum of values

We can now multiply attention weights by corresponding value vactor



4.5) multi-head Attention

Its like

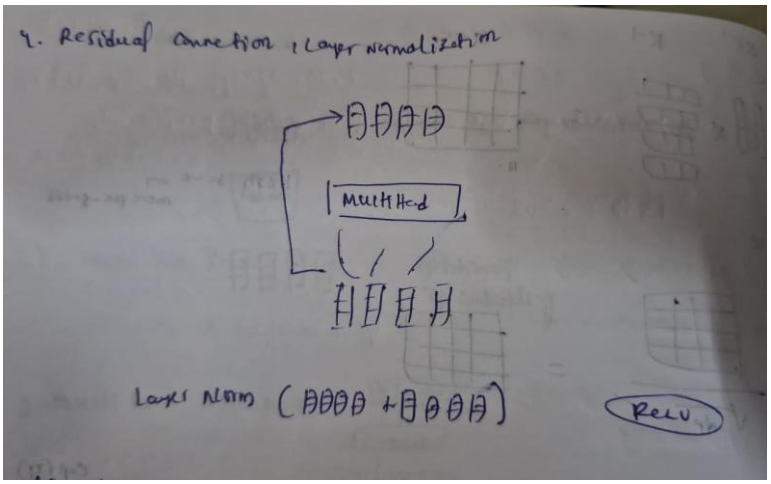
Self-attention head 1 + Self-attention head 2 + ……..+ Self-attention head n

Here we also include the another one after this step

6. Residual connection, layer normalization

Residual Connection: A shortcut connection that adds the input of a layer to its output, helping gradients flow more easily during backpropagation.

Layer Normalization: A technique that normalizes the inputs of each layer to have a mean of zero and variance of one, improving model stability and convergence.

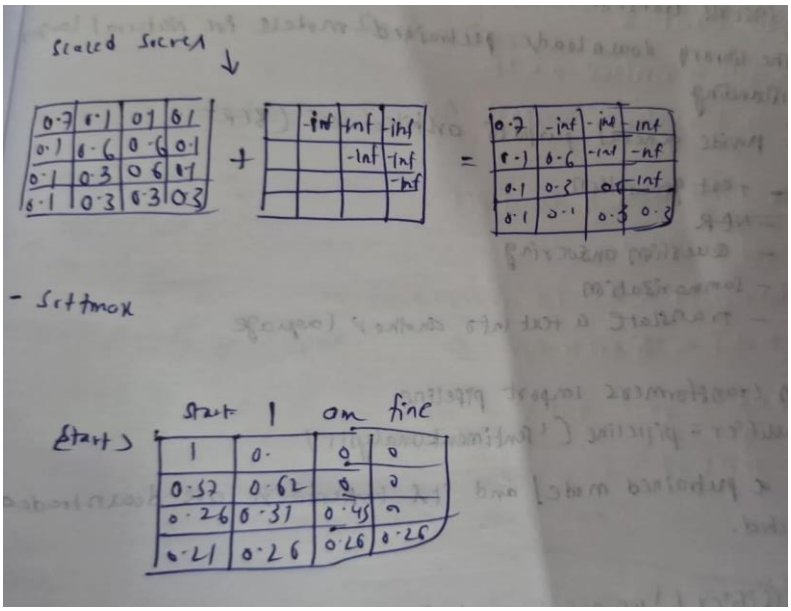


Decoder

The operations are same like Encoder but here we use Masked Multi-Headed Attention

i) Masked Multi-Head Attention

When we are computing the words, we should not access to the future words that's why we use this.



Finally, we use linear classifier, At the output of the decoder in the "Attention is All You Need" paper, a **linear classifier** is applied to transform the decoder's output vectors into logits for each token in the vocabulary. This is done by passing the decoder's output through a linear (fully connected) layer followed by a SoftMax activation to predict the probability distribution over the vocabulary, enabling the selection of the most likely next token.

Here are the top 5 most influential and widely used transformer models:

BERT (Bidirectional Encoder Representations from Transformers)

A milestone in NLP, BERT is pre-trained on large corpora and excels in tasks like question answering, sentiment analysis, and text classification.

GPT (Generative Pre-trained Transformer)

Known for its text generation capabilities, GPT powers conversational AI systems and content creation tools.

ViT (Vision Transformer)

Pioneered the use of transformers for image recognition by splitting images into patches and applying attention mechanisms.

T5 (Text-to-Text Transfer Transformer)

Frames every NLP problem as a text-to-text task, offering a unified approach to diverse NLP challenges.