Transformers

# Overview

Motivation: Attention mechanism facilitates longer sequences of dependencies  
Architecture: An encoder-decoder network

# Encoder Steps & Layers

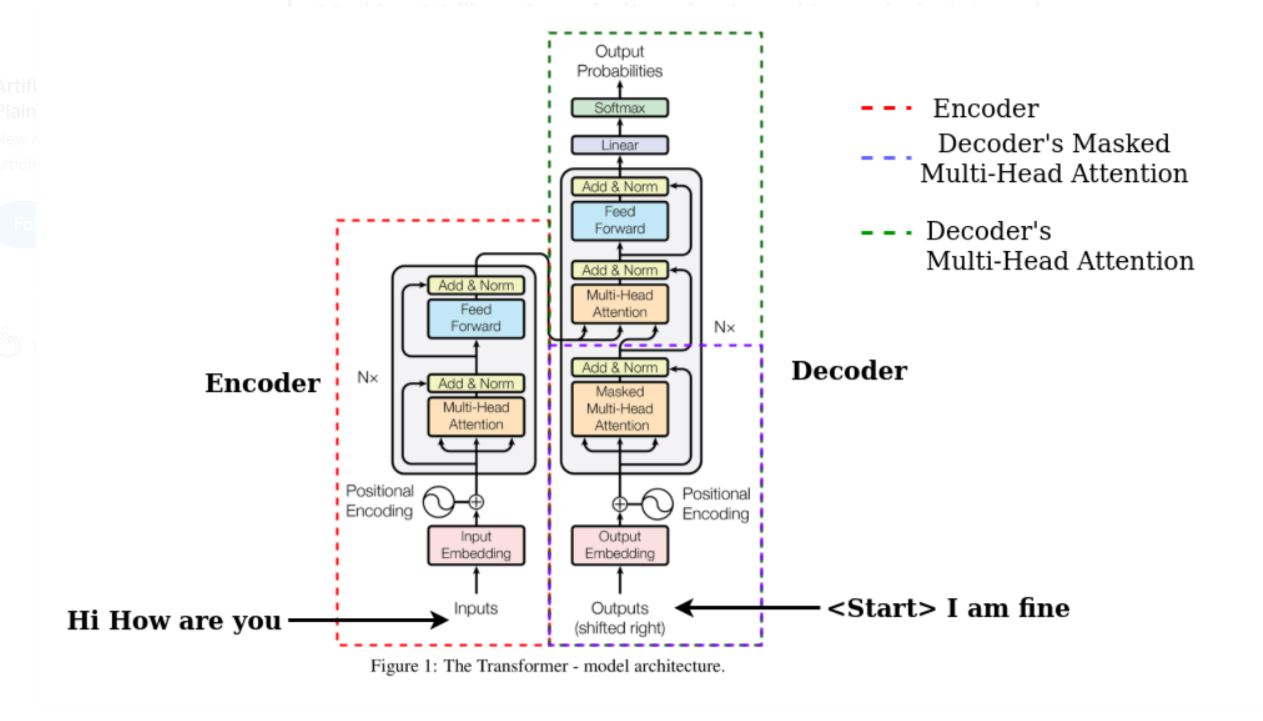
1. Input sentence
2. Encode each word (via an embedding)
3. Create a positional encoding (using sin & cos)
4. Add input and positional 🡺 positional input encoding
5. Create Query, Key and Value Vectors
   1. Self-Attention mechanism
   2. Associate each word with every other word
6. Dot Product of Query and Keys
   1. Score matrix
   2. Measures focus between words
7. Scaling Down Score Matrix (SM)
   1. Softmax (SM / √d\_k) = Attention Weights matrix
   2. d\_k is dimension of keys and queries.
8. Drowning Out Irrelevant Words using Attention Weights
   1. Dot Product of Attention Weights and Values
   2. Produces Output Matrix
9. Feed Forward Neural Network
   1. Output Matrix 🡪 FFN 🡪 Encoder Output
   2. One encoder or one single attention block. (single head)
10. Multi-Head Attention
    1. Copy Q, K and V vectors across different N heads
    2. Multiple Q, K, V with separate self-attention mechanisms
    3. Increases representational power

# Encoder Summary

Multi-Head Attention Module in transformer network computes the attention weights for the inputs and produces output vector with encoded information of how each word should attend to all other words in the sequence.

# Decoder Steps & Layers

In a decoder block, there are two multi-head attention modules. In the first masked multi-head attention, we pass in the decoder input and in the second multi-head attention we pass in encoder’s output and the first multi-head attention module’s output.



It should be noted that decoder is an auto-regressive model meaning it predicts future behavior based on past behavior.

1. Creating Value Vectors
   1. Create the value vectors using decoder input
   2. Generate attention weights.
   3. Masked Multi-Head Attention
   4. Masking is added before the Softmax and after scaling the scores.

In summary, the first Masked Multi-Head Attention creates a masked output vector with information on how the model should attend on the decoders input.

1. Decoder’s Multi-Head Attention Output
   1. Multi-Head attention matches the encoder’s output with decoder output (masked output) allowing the decoder to decide which encoder output is relevant and focus on.
   2. Then output from second multi-head attention is passed through pointwise FFNN for further processing.
   3. The output of FFNN through Linear Layer, which acts as Classifier Layer
   4. Each word of vocab has a probability score after going through Softmax function.
   5. The max. probability is our predicted word. This word is again sent back to lists of decoder inputs.
   6. This process continues until the decoder generates the <END> token.