**What is transformer? Why we need transformer? How its works?**

**What is transformer?**

1. A **Transformer** is a deep learning architecture introduced in the seminal paper "Attention is All You Need" (2017) by Vaswani et al.
2. It revolutionized natural language processing (NLP) and other fields by enabling highly efficient and scalable sequence-to-sequence tasks
3. Transformers rely on a self-attention mechanism to weigh the importance of different parts of the input sequence. This allows them to process long-range dependencies more efficiently and effectively.

**Problems with Earlier Architectures and Transformer's Solution?**

1. RNNs, LSTMs, and GRUs process data sequentially.
2. While LSTMs and GRUs mitigate the vanishing gradient problem compared to vanilla RNNs, they are still susceptible when handling very long sequences.

**How Transformer works?**

**Encoder Components**:

1. Self-Attention Mechanism
2. Position-Wise Feed-Forward Network (FFN):
3. Residual connections and layer normalize

**Decoder Components**:

1. Masked Self-Attention
2. Cross attention
3. Encoder-Decoder Attention

Summary of Internal Stages in the Transformer:

1. Tokenization → Embedding → Positional Encoding.
2. Self-attention (scaled dot-product).
3. Multi-head attention (parallel attention heads).
4. Feedforward neural network.
5. Residual connections + Layer normalization.
6. Stack NNN layers (deep representation learning).
7. (If applicable) Encoder-decoder attention mechanism.
8. Masking for padding and causal relationships.
9. Output logits → softmax → token probabilities.
10. Training with cross-entropy loss.
11. **Tokenization:** Process of breaking down a larger piece of text, such as a sentence or paragraph, into smaller
12. **Token Embedding**: Each word in the input sequence is converted into a dense vector of fixed dimensionality. These embeddings encode semantic information about words.
13. **Positional Encoding**: Since the Transformer does not process data sequentially, positional encodings are added to embeddings to provide information about token positions in the sequence. This is done using sine and cosine functions of different frequencies.
14. **Self-attention mechanism** the self-attention mechanism in transformers is a key component that allows the model to consider the entire sequence of input data at once, rather than processing it sequentially. This mechanism helps capture relationships between words or tokens in a sentence regardless of their positions in the sequence, which is particularly useful for tasks like language modelling, translation, or text generation.
15. **Multi Head Attention:** It extends self-attention by employing multiple parallel attention mechanisms for better feature representation.

It splits the attention mechanism into multiple "heads," where each head learns distinct representations or attends to different positions in the input.

1. **Feedforward Neural Network (FNN):** A Feedforward Neural Network (FNN) is a type of artificial neural network where information flows in one direction — from the input layer, through one or more hidden layers, to the output layer.

The Feedforward Neural Network in Transformers acts as a refinement stage, enhancing token embeddings after self-attention. It provides non-linearity, expressive transformations, and richer representations, enabling the Transformer to model complex relationships in data.

1. **Residual Connections and Layer Normalization:** Both residual connections and layer normalization are crucial components of the Transformer architecture. They improve training stability, enable deeper architectures, and ensure effective gradient flow.
2. **Stacking Layers:** Both encoder and decoder are built from a stack of identical layers
3. **Masked Self-Attention**: Masked Self-Attention ensures that, during training or generation, each token in a sequence can only attend to itself and previous tokens, not future ones. This prevents the model from "seeing" future tokens, maintaining causality in sequence prediction tasks.
4. **Encoder-Decoder Attention**: Allows the decoder to attend to encoder outputs, integrating context from the input sequence.

**Training Details:**

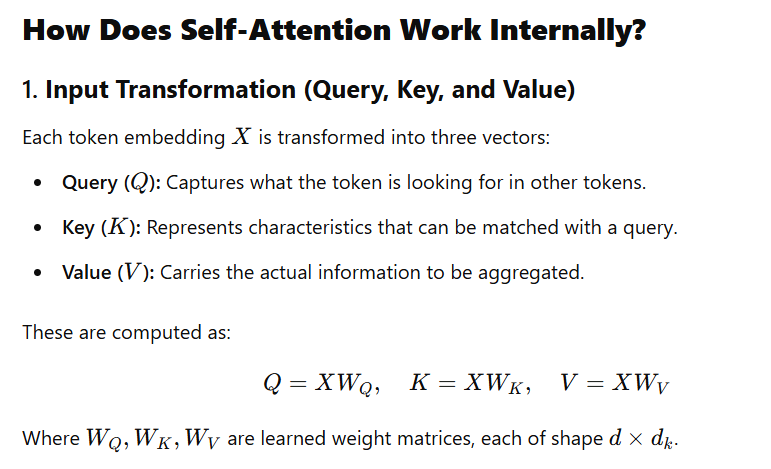
1. **Optimization**: Trained using the Adam optimizer with a novel learning rate schedule.
2. **Loss Function**: Cross-entropy loss is used, with teacher forcing during training.
3. **Regularization**:
   1. **Dropout** in sub-layers.
   2. **Label smoothing** to prevent overconfidence.

**Transformer Interview Questions:**

1. What is a Transformer model, and how does it differ from traditional RNNs and LSTMs?
2. Explain the self-attention mechanism and how it works in a transformer.
3. What is the role of positional encoding in transformers? Why is it needed?
4. Can you describe the architecture of a transformer model, including the components of the encoder and decoder?
5. What are the advantages of using transformers over recurrent neural networks (RNNs) for sequence modelling?
6. What is multi-head attention in transformers, and why is it useful?
7. How does a transformer handle long-term dependencies in sequences more effectively than RNNs or LSTMs?
8. What is the purpose of the "feed-forward neural network" layer in each transformer block?
9. What are the hyperparameters in a transformer model, and how do they affect performance?
10. How do transformers handle variable-length input sequences during training and inference?
11. Explain the concept of "layer normalization" and its significance in transformer models.
12. What is the difference between BERT, GPT, and T5 architectures in the context of transformers?
13. How do transformers improve parallelization compared to RNN-based architectures?
14. What are the challenges of training large transformer models, and how are they addressed (e.g., with techniques like attention masking or gradient checkpointing)?
15. What are some real-world applications of transformer-based models in NLP and other fields?

A white paper with black text

Description automatically generated



A close up of black text

Description automatically generated

A white background with black text

Description automatically generated

A close-up of a white background

Description automatically generated

A close-up of a white background

Description automatically generated

A white background with black text

Description automatically generated

A white background with black text

Description automatically generated

A black and white text with black text

Description automatically generated

**Self-attention working flow:**

Q K V vector calculation 🡪 Attention score 🡪 Scaling Scores -🡪 SoftMax 🡪Weighted Sum

**What is use of multi head attention?**

Multi-Head Attention takes this **self-attention process** and applies it **in parallel**

Each "head" operates independently and attends to different aspects of the input.

Each head independently captures unique relationships in the data, such as context, positional relevance, or dependencies between tokens.

**What is use of feed forward network [FFN]?**

The FFN introduces non-linearity to the model. It helps to learn complex, non-linear relationships in the data.

**What is use of Residual Connections?**

Residual Connections gradient flow, stabilizes training, and enables the **effective training of very deep networks** with better convergence and improved performance.

Mitigate the Vanishing Gradient Problem

Smooth Optimization

**What is use of Layer normalization?**

Layer normalization in transformers standardizes the inputs to each layer, stabilizing and accelerating training by ensuring consistent scaling of features. It helps prevent exploding or vanishing gradients

Faster Convergence can achieve from layer normalization.

**What are the hyperparameters in a transformer model, and how do they affect performance?**

1. Number of encoder and decoder layers in the transformer
2. Dimensionality of Embeddings size
3. Number of Attention Heads
4. Feedforward Network Size
5. Learning rate, batch size, weight initialization
6. Dropout Rate, Label smoothing
7. Vocabulary Size
8. Training Epochs
9. How do transformers handle variable-length input sequences during training and inference?

|  |  |
| --- | --- |
| * 1. **Positional Encoding** | Encodes position information for variable-length input. |

|  |  |
| --- | --- |
| * 1. **Padding** | Ensures uniform sequence lengths in a batch. |

How encoder and decoder work in transformer?

A white background with black text

Description automatically generated