TRANSFORMERS

Transformers are a type of deep learning model that utilizes self-attention mechanisms to process and generate sequences of data efficiently. They capture long-range dependencies and contextual relationships making them highly effective for tasks like language modeling, machine translation and text generation. Transformer model is built on encoder-decoder architecture where both the encoder and decoder are composed of a series of layers that utilize self-attention mechanisms and feed-forward neural networks. This architecture enables the model to process input data in parallel making it highly efficient and effective for tasks involving sequential data.

- The encoder processes input sequences and creates meaningful representations.
- The decoder generates outputs based on encoder representations and previously predicted tokens.

The encoder and decoder work together to transform the input into the desired output such as translating a sentence from one language to another or generating a response to a query.

1. Encoder

The primary function of the encoder is to create a high-dimensional representation of the input sequence that the decoder can use to generate the output. Encoder consists of multiple layers and each layer is composed of two main sub-layers:

1. Self-Attention Mechanism: This sub-layer allows the encoder to weigh the importance of different parts of the input sequence differently to capture dependencies regardless of their distance within the sequence.
2. Feed-Forward Neural Network: This sub-layer consists of two linear transformations with a ReLU activation in between. It processes the output of the self-attention mechanism to generate a refined representation.

Layer normalization and residual connections are used around each of these sub-layers to ensure stability and improve convergence during training.

2. Decoder

Decoder in transformer also consists of multiple identical layers. Its primary function is to generate the output sequence based on the representations provided by the encoder and the previously generated tokens of the output.

Each decoder layer consists of three main sub-layers:

1. **Masked Self-Attention Mechanism**: Similar to the encoder's self-attention mechanism but its main purpose is to prevent attending to future tokens to maintain the autoregressive property (no cheating during generation).
2. **Encoder-Decoder Attention Mechanism**: This sub-layer allows the decoder to focus on relevant parts of the encoder's output representation. This allows the decoder to focus on relevant parts of the input, essential for tasks like translation.
3. **Feed-Forward Neural Network**: This sub-layer processes the combined output of the masked self-attention and encoder-decoder attention mechanisms.

Need For Transformers Model in Machine Learning

Transformer Architecture uses self-attention to transform one whole sentence into a single sentence. This is useful because older models work step by step and it helps overcome the challenges seen in models like RNNs and LSTMs. Traditional models like RNNs (Recurrent Neural Networks) suffer from the vanishing gradient problem which leads to long-term memory loss. RNNs process text sequentially meaning they analyze words one at a time.

Core Concepts of Transformers

1. Self Attention Mechanism

The self attention mechanism allows transformers to determine which words in a sentence are most relevant to each other. This is done using a scaled dot-product attention approach:

Each word in a sequence is mapped to three vectors:

- **Query (Q)**
- **Key (K)**
- **Value (V)**

Attention scores are computed as: $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

These scores determine how much attention each word should pay to others.

2. Positional Encoding

Unlike RNNs, transformers lack an inherent understanding of word order since they process data in parallel. To solve this problem Positional Encodings are added to token embeddings providing information about the position of each token within a sequence.

3. Multi-Head Attention

Instead of one attention mechanism, transformers use multiple attention heads running in parallel. Each head captures different relationships or patterns in the data, enriching the model's understanding.

4. Position-wise Feed-Forward Networks

The Feed-Forward Networks consist of two linear transformations with a ReLU activation. It is applied independently to each position in the sequence.

This transformation helps refine the encoded representation at each position.

5. Encoder-Decoder Architecture

The **encoder-decoder** structure is key to transformer models. The encoder processes the input sequence into a vector, while the decoder converts this vector back into a sequence. Each encoder and decoder layer includes self-attention and feed-forward layers. In the decoder, an encoder-decoder attention layer is added to focus on relevant parts of the input.

For example, a French sentence "Je suis étudiant" is translated into "I am a student" in English.

The encoder consists of multiple layers (typically 6 layers). Each layer has two main components:

- **Self-Attention Mechanism:** Helps the model understand word relationships.
- **Feed-Forward Neural Network:** Further transforms the representation.

The decoder also consists of 6 layers but with an additional encoder-decoder attention mechanism. This allows the decoder to focus on relevant parts of the input sentence while generating output.

How Transformers Work

1. Input Representation

The first step in processing input data involves converting raw text into a format that the transformer model can understand. This involves tokenization and embedding.

- **Tokenization**: The input text is split into smaller units called tokens, which can be words, sub words or characters. Tokenization ensures that the text is broken down into manageable pieces.
- **Embedding**: Each token is then converted into a fixed-size vector using an embedding layer. This layer maps each token to a dense vector representation that captures its semantic meaning.
- **Positional encodings** are added to these embeddings to provide information about the token positions within the sequence.

2. Encoder Process in Transformers

- **Input Embedding**: The input sequence is tokenized and converted into embeddings with positional encodings added.
- **Self-Attention Mechanism**: Each token in the input sequence attends to every other token to capture dependencies and contextual information.
- **Feed-Forward Network**: The output from the self-attention mechanism is passed through a position-wise feed-forward network.
- **Layer Normalization and Residual Connections**: Layer normalization and residual connections are applied.

3. Decoder Process

- **Input Embedding and Positional Encoding**: The partially generated output sequence is tokenized and embedded with positional encodings added.
- **Masked Self-Attention Mechanism**: The decoder uses masked self-attention to prevent attending to future tokens ensuring that the model generates the sequence step-by-step.
- **Encoder-Decoder Attention Mechanism**: The decoder attends to the encoder's output allowing it to focus on relevant parts of the input sequence.
- **Feed-Forward Network**: Similar to the encoder the output from the attention mechanisms is passed through a position-wise feed-forward network.
- **Layer Normalization and Residual Connections**: Similar to the encoder Layer normalization and residual connections are applied.

4. Training and Inference

- Transformers are trained with teacher forcing, where the correct previous tokens are provided during training to predict the next token. Their encoder-decoder architecture combined with multi-head attention and feed-forward networks enables highly effective handling of sequential data.
- Transformers have transformed deep learning by using self-attention mechanisms to efficiently process and generate sequences capturing long-range dependencies and contextual relationships. Their encoder-decoder architecture combined with multi-head attention and feed-forward networks enables highly effective handling of sequential data.

References

GeeksforGeeks. (2025, October 18). *Architecture and working of transformers in deep learning*. GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning/architecture-and-working-of-transformers-in-deep-learning/