Transformer Architecture Overview

Code is available at [https://github.com/khood5/ADVANCED_DEEP_LEARNING_CS_69995_005_Homework3.git]

The following is an overview of the code provided in this GitHub repo. The code is very similar, but not identical to, the one written by Umar Jamil in his video tutorial on transformers [https://www.youtube.com/watch?v=ISNdQcPhsts&ab_channel=UmarJamil]

While the model was written by following the tutorial due to a bug I copied the "**`positionEncoding`**" from his code. I left my original one in as a comment if you can find the bug I would like to know what it is. Apparently in mine the "**`positionEncodingMatrix`**" tensor never makes it to the same device the rest of the model does. The cause for this is unknown to me. I only describe the methods of the model due to space limitations of this report.

Key Components

1. Input Embedding: `textInputEmbedding`

The input embedding layer converts input tokens into continuous vector representations. This is achieved using an embedding matrix, which is trainable during the training process.

2. Positional Encoding: `positionEncoding`

To provide positional information to the model, positional encodings are added to the input embeddings. This allows the model to understand the order of tokens in a sequence. The positional encoding is created using sine and cosine functions with different frequencies.

3. Normalization Layer: `addNormLayer`

This layer performs normalization on the input by centering and scaling the values. It includes learnable parameters (alpha and beta) to adjust the normalized values.

4. Feedforward Layer: `ffLayer`

The feedforward layer consists of two linear transformations with a ReLU activation in between. This layer introduces non-linearity to the model.

5. Multi-Head Attention Layer: `multiHeadAttentionLayer`

This layer performs multi-head self-attention, allowing the model to focus on different parts of the input sequence simultaneously. It includes separate linear transformations for the query, key, and value projections. The attention scores are scaled and passed through a softmax function.

6. Residual Connection: `residualConnection`

The residual connection allows the model to retain information from the input by adding the original input to the output of a sub-layer. This is followed by layer normalization.

7. Encoder Layer: `encoderLayer`

The encoder layer consists of a self-attention block and a feedforward block, each followed by a residual connection. This layer is repeated in the encoder.

8. Decoder Layer: `decoderLayer`

The decoder layer extends the encoder layer by adding a cross-attention block that attends to the encoder's output. It also includes a residual connection and layer normalization.

9. Encoder: `encoder`

The encoder is composed of multiple encoder layers stacked on top of each other.

10. Decoder: `decoder`

The decoder is composed of multiple decoder layers stacked on top of each other.

11. Projection Layer: `projectionLayer`

The projection layer transforms the decoder's output into probabilities over the target vocabulary using a linear transformation followed by a softmax activation.

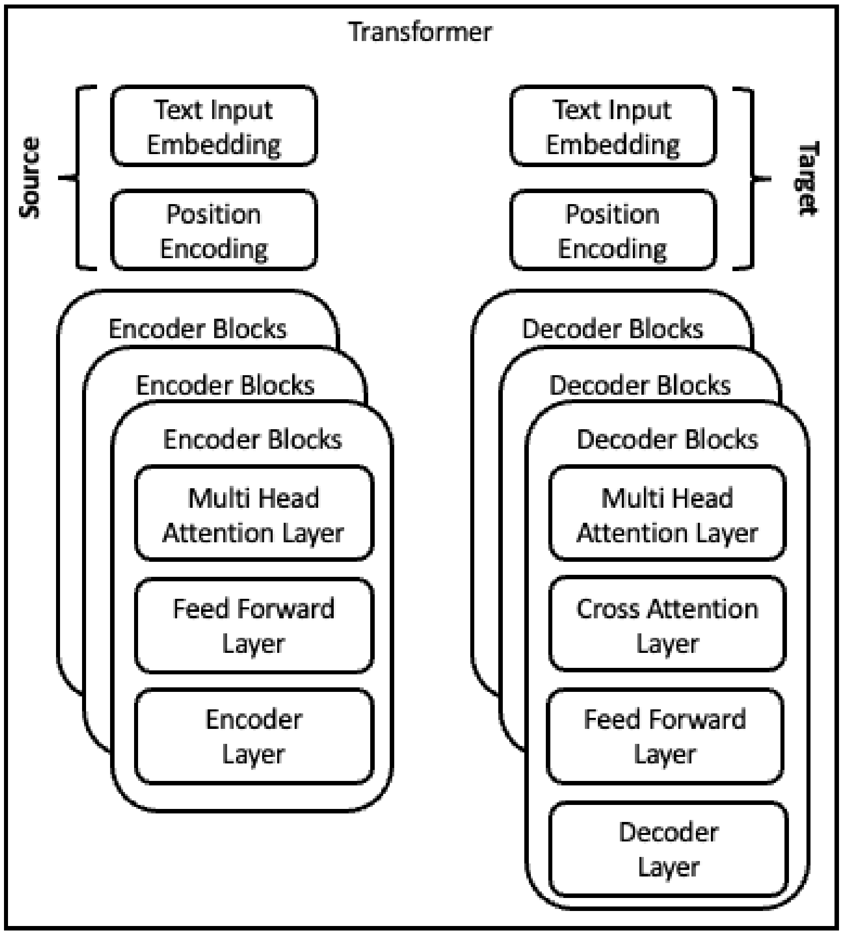
12. Transformer Model: `transformer`

The main Transformer model is composed of an encoder, a decoder, input embeddings, positional encodings, and a projection layer.

Appendix

Training and testing where done on the same dataset Umar Jamil used in his video. I have added a typescript file "**typescript**" that shows a successful run of training of the transformer.

System diagram


Transformer
Source
Text Input Embedding
Position Encoding
Target
Text Input Embedding
Position Encoding
Encoder Blocks
Encoder Blocks
Encoder Blocks
Multi Head Attention Layer
Feed Forward Layer
Encoder Layer
Decoder Blocks
Decoder Blocks
Decoder Blocks
Multi Head Attention Layer
Cross Attention Layer
Feed Forward Layer
Decoder Layer