TransformerDecoder layer

`TransformerDecoder` class

[source]

```
keras_nlp.layers.TransformerDecoder(
    intermediate_dim,
    num_heads,
    dropout=0,
    activation="relu",
    layer_norm_epsilon=1e-05,
    kernel_initializer="glorot_uniform",
    bias_initializer="zeros",
    name=None,
    **kwargs
)
```

Transformer decoder.

This class follows the architecture of the transformer decoder layer in the paper Attention is All You Need. Users can instantiate multiple instances of this class to stack up a decoder.

This layer will always apply a causal mask to the decoder attention layer. This layer will correctly compute an attention mask from an implicit Keras padding mask (for example, by passing `mask_zero=True` to a `keras.layers.Embedding` layer). See the Masking and Padding guide for more details.

This layer can be called with either one or two inputs. The number of inputs must be consistent across all calls. The options are as follows: `layer(decoder_sequence)`: no cross-attention will be built into the decoder block. This is useful when building a "decoder-only" transformer such as GPT-2. `layer(decoder_sequence, encoder_sequence)`: cross-attention will be built into the decoder block. This is useful when building an "encoder-decoder" transformer, such as the original transformer model described in Attention is All You Need.

Arguments

- **intermediate_dim**: int, the hidden size of feedforward network.
- **num_heads**: int, the number of heads in MultiHeadAttention.
- **dropout**: float, defaults to 0. the dropout value, shared by MultiHeadAttention and feedforward network.
- **activation**: string or `keras.activations`, defaults to "relu". the activation function of feedforward network.
- **layer_norm_epsilon**: float, defaults to 1e-5. The eps value in layer normalization components.
- **kernel_initializer**: string or `keras.initializers` initializer, defaults to "glorot_uniform". The kernel initializer for the dense and multiheaded attention layers.
- **bias_initializer**: string or `keras.initializers` initializer, defaults to "zeros". The bias initializer for the dense and multiheaded attention layers.
- **name**: string, defaults to None. The name of the layer.
- ****kwargs**: other keyword arguments.

Examples

```
# Create a single transformer decoder layer.
decoder = keras_nlp.layers.TransformerDecoder(
    intermediate_dim=64, num_heads=8)

# Create a simple model containing the decoder.
decoder_input = keras.Input(shape=[10, 64])
encoder_input = keras.Input(shape=[10, 64])
output = decoder(decoder_input, encoder_input)
model = keras.Model(inputs=[decoder_input, encoder_input],
    outputs=output)

# Call decoder on the inputs.
decoder_input_data = tf.random.uniform(shape=[2, 10, 64])
encoder_input_data = tf.random.uniform(shape=[2, 10, 64])
decoder_output = model([decoder_input_data, encoder_input_data])
```

References

- Vaswani et al., 2017

`call` method [source]

```
TransformerDecoder.call(
    decoder_sequence,
    encoder_sequence=None,
    decoder_padding_mask=None,
    decoder_attention_mask=None,
    encoder_padding_mask=None,
    encoder_attention_mask=None,
)
```

Forward pass of the TransformerDecoder.

Arguments

- **decoder_sequence**: a Tensor. The decoder input sequence.
- **encoder_sequence**: a Tensor. The encoder input sequence. For decoder only models (like GPT2), this should be left None. Once the model is called once without an encoder_sequence, you cannot call it again with encoder_sequence.
- **decoder_padding_mask**: a boolean Tensor, the padding mask of decoder sequence, must of shape [batch_size, decoder_sequence_length].
- **decoder_attention_mask**: a boolean Tensor. Customized decoder sequence mask, must of shape [batch_size, decoder_sequence_length, decoder_sequence_length].
- **encoder_padding_mask**: a boolean Tensor, the padding mask of encoder sequence, must of shape [batch_size, encoder_sequence_length].
- **encoder_attention_mask**: a boolean Tensor. Customized encoder sequence mask, must of shape [batch_size, encoder_sequence_length, encoder_sequence_length].

Returns

A Tensor of the same shape as the `decoder_sequence`.
