Neural Machine Translation

Definition

Using Neural Networks to translate text from a human language to another.

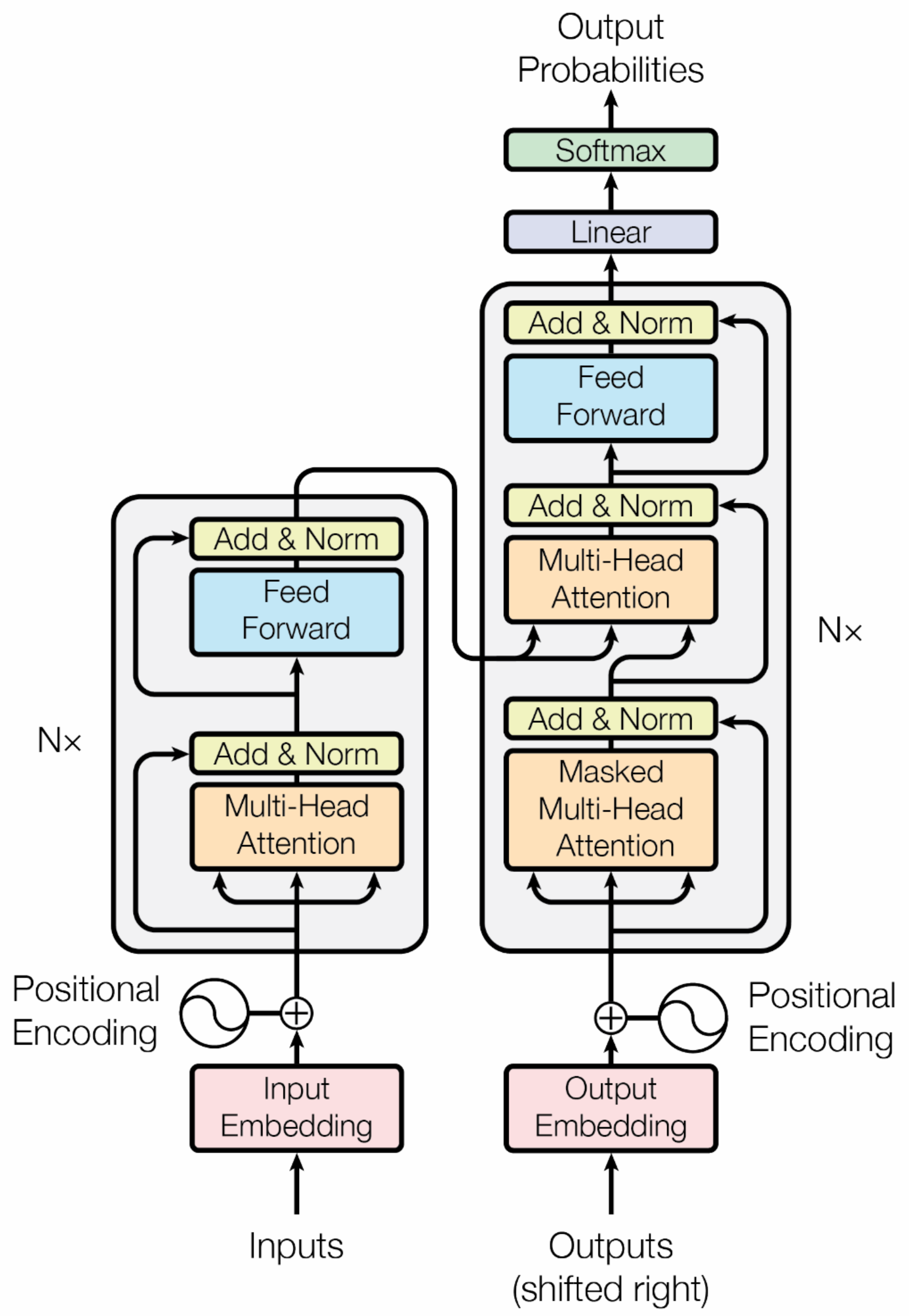
Transformers

Attention

Mapping a query and a set of key-value pairs to an output.
The output is computed as a weighted sum of the values.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}V)$$

Mo


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Multi-head self attention mechanism,

Use H attention functions instead of 1.

$$Multihead(Q,K,V) = contact(head_1,\ldots,head_h)W^O where head_i = attention(QW_i^Q, KW_i^K, VW_i^V)$$

Application of multi-head self-attention mechanism ???

- In "Encoder-Decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence. This mimics the typical encoder-decoder attention mechanisms in sequence-to-sequence models.
- The encoder contains self-attention layers. In a self-attention layer, all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder.
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including the position. We need to prevent leftward information flow in the decoder to preserve the auto-regressive property. We implement this inside of scaled Dot-Product attention by masking out (setting to $-\infty$) all values in the input of the softmax which correspond to illegal connection. See figure \ref{fig:scaled-dot-product-attention}. %figure 2 in transformer

Position-wise fully connected feed forward network

2 linear transformations with ReLU in between$FFN(x) = max(0, xW_1 + b_1) + W_2 + b_2$

Embeddings and Softmax

Similar to other sequence transduction models, this model uses learned embeddings to convert the input tokens and output tokens to vectors of dimension D_{model}. We also use the usual learned linear transformation and softmax function to convert the decoder output to predicted next-token probabilities. In this

Positional Encoding

Because what we have learned till now doesn't reserve position of input token, we use positional encoding to provide it.$PE_{(POS,2i)} = sin(POS/10000^{2i/d_{model}})$∫

Language Modeling

Bidirectional Encoder Representations From Transformers (BERT)

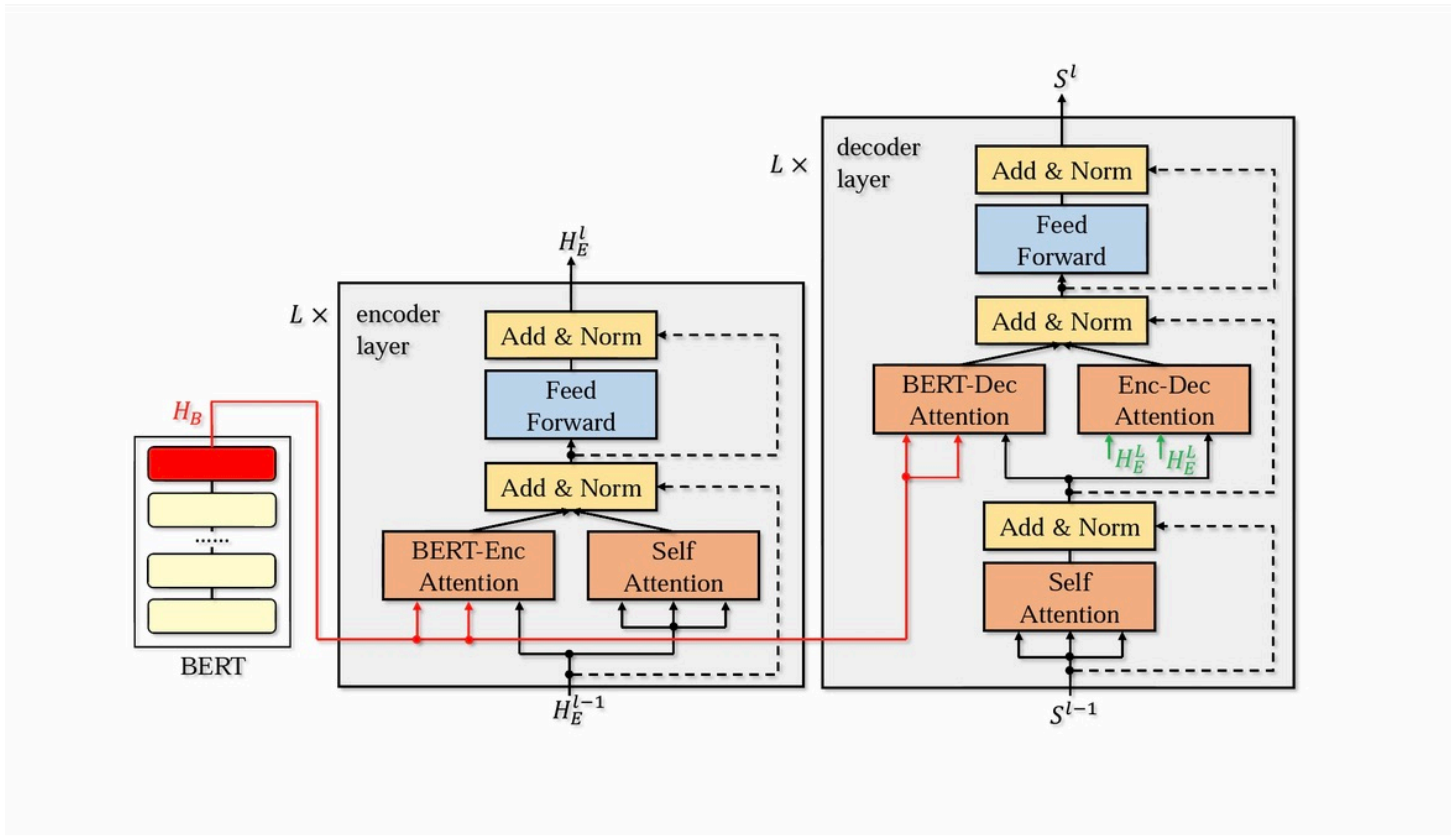
Technique for NLP pre-training

BERT-fused

BERT+Transformer

- BERT+Encoder+decoder
- BERT+Decoder

BERT-fused Model



Step 1

For input x, BERT makes representation HB

$$H_B = BERT(x)$$

Step 2

$$\tilde{h}_i^l = \frac{1}{2}(attn_S(h_i^{l-1}, H_E^{l-1}, H_E^{l-1}) + attn_B(h_i^{l-1}, H_B, H_B)), \forall i \in [l_x]$$

Step 3

$\hat{S}_t^l = attn_s(S_t^{l-1}, S_{<t-1}^{l-1}, S_{<t-1}^{l-1});$

Our NMT model

- BERT Base (cased & multilingual uncased)
- 0 encoder layers
- 6 decoder layers

Measuring performance of NMT system

- Label smoothing cross entropy(training)
- Sacrebleu (testing the model)

Label Smooth Cross Entropy

$$L = -\sum_{i=1}^{n} [(1-\epsilon)H_i(p, q_\theta) + \epsilon H_i(U, Q_\theta)]$$