SIMPLIFIED TRANSFORMER-ENCODER ARCHITECTURE

⚠ Missing components:
- Layer normalization
- Skip connections
- Position-wise feed-forward layer

~ BERT sizes

(max. 512 tokens)
seq_len
(256 - 512)
batch_size
one float, one dimension of embed. vector
embed_dim
(768 or 1024)
base large
one token represented as embeding vector
one sequence of tokens

=

embed_dim
seq_len
X input

⊕

embed_dim
seq_len
P position embeddings
(can be absolute, relative, or learned)

→

embed_dim
seq_len
Z = X + P
input embeddings ⊕ position encoding

(One) Self-Attention Head

Z

head_dim
embed_dim
W_Q
note W_Qs are learned and independent from seq_len!
$Q = ZW_Q$
(embed_dim/M, M=12)
head_dim < embed_dim
seq_len
Q queries

$K = ZW_K$
head_dim
seq_len
K keys

$V = ZW_V$
head_dim
seq_len
V values

① projections to Q, K, V

② Attention Layer output

$Y(Q,K,V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{\text{head\_dim}}}\right) \cdot V$

Dot-product similarity between all projected sequence tokens! seq_len x seq_len

highest similarity value is exponentially highlighted, lowers decreased

head_dim
seq_len

The effect of a linear matrix product is that all projected tokens in the sequence V are summed → thus, the order info is lost → positional encoding is needed!

dim(X) = dim(Z) = dim(Y)

same Z processed in parallel

Z

$W_Q \rightarrow Q$
$W_K \rightarrow K$
$W_V \rightarrow V$
$\rightarrow Y(Q,K,V)$

head_dim head_dim head_dim
seq_len
embed_dim : all attention head outputs are concatenated!

=

contextualized embeddings
hidden_dim = embed_dim
seq_len

linear or feed-forward projection with a dense an learnable W_O

x W_O
embed_dim x embed_dim

Y

(One) Multi-head Attention

↳ We have N multi-head attention blocks stacked serially
(N = 12 (base) or 16 (large) in BERT)

(M self-attention layers)
(M=12 in BERT)