Lecture 9: Transformers

Presenter: Chu Đình Đức

Lecture Plan

1. Impact of Transformers on NLP (and ML more broadly)
2. From Recurrence (RNNs) to Attention-Based NLP Model
3. Understanding the Transformer Model
4. Drawbacks and Variants of Transformers

1. Impact of Transformers on NLP

- We learn that attention dramatically improves the performance of RNN
- Today, we will take this one step further and ask *is attention all we need*?

Attention Is All You Need

Ashish Vaswani[*] Google Brain avaswani@google.com

Noam Shazeer[*] Google Brain noam@google.com

Niki Parmar[*] Google Research nikip@google.com

Jakob Uszkoreit[*] Google Research usz@google.com

Llion Jones[*] Google Research llion@google.com

Aidan N. Gomez[*] [†] University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser[*] Google Brain lukaszkaiser@google.com

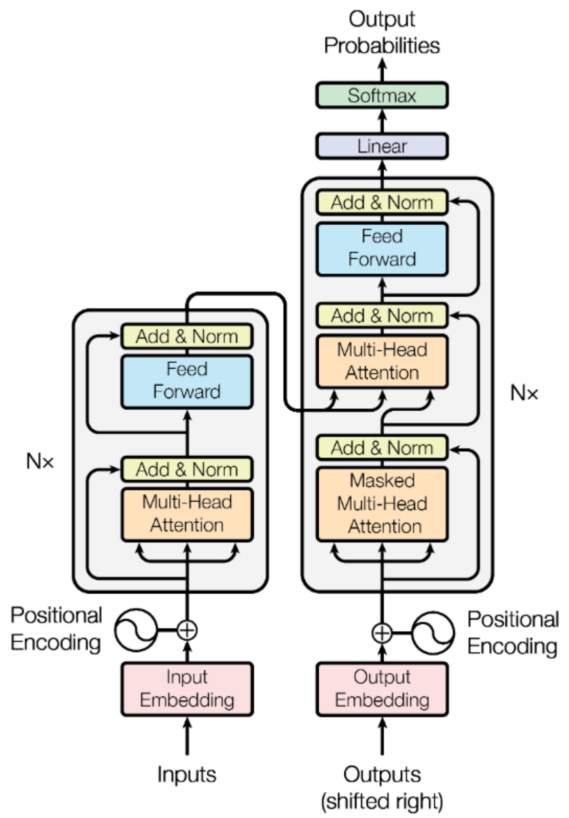
Illia Polosukhin[*] [‡] illia.polosukhin@gmail.com

2. From RNNs to Attention-Based NLP Models

Issues with recurrent models

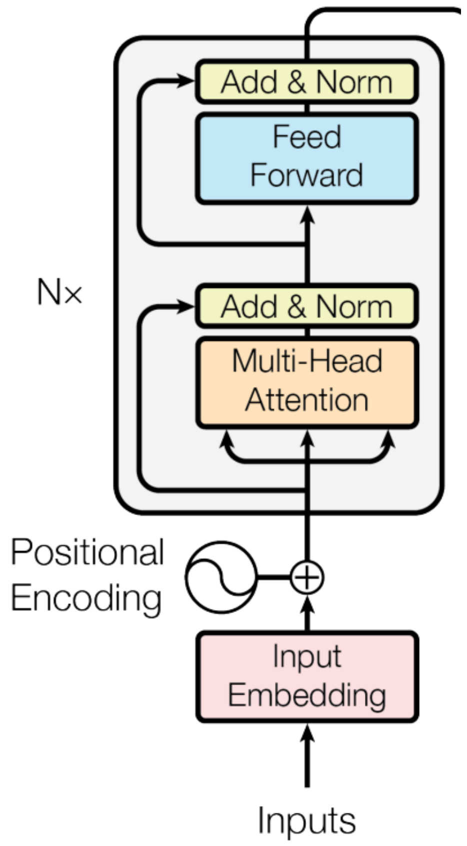
- Linear interaction distance
    - Hard to learn long-distance dependencies (because gradient problems)
    - We already know sequential structure doesn't tell the whole story
- Lack of parallelizability
- Solution: (self) attention

3. Understanding the Transformer Model



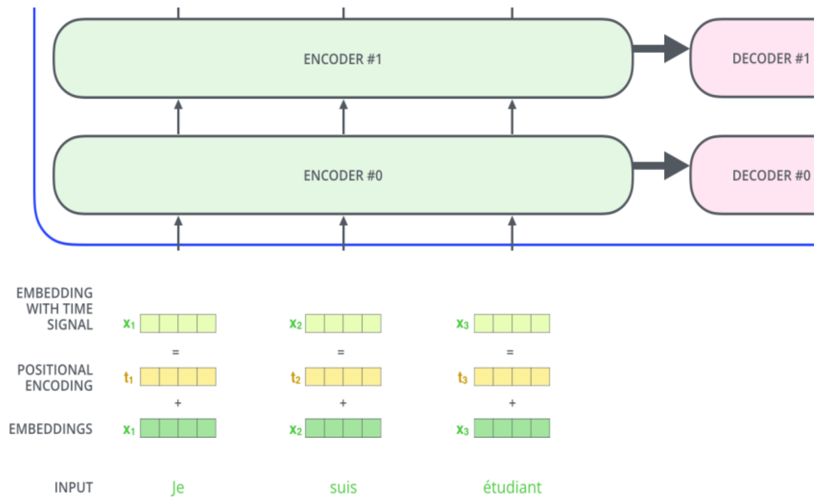
3.1 Encoder Conclusion

- The encoder is composed of a stack of N = 6 identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network
- We employ a residual connection around each of the two sub-layers, followed by layer normalization. That is, the output of each sub-layer is *LayerNorm(x+Sublayer(x))*



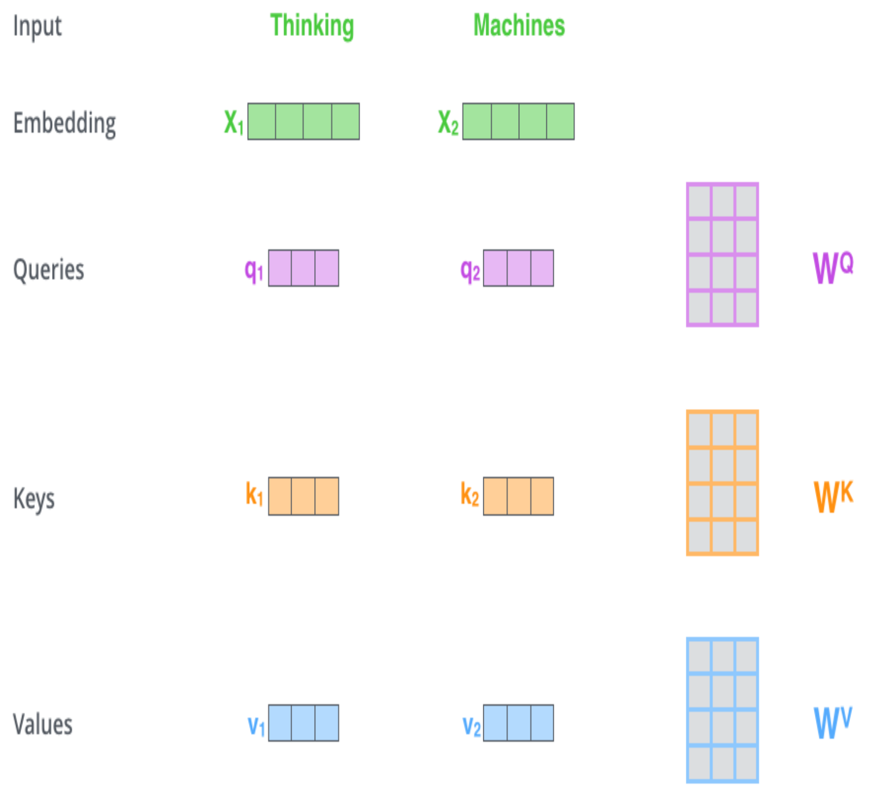
Positional Encoding

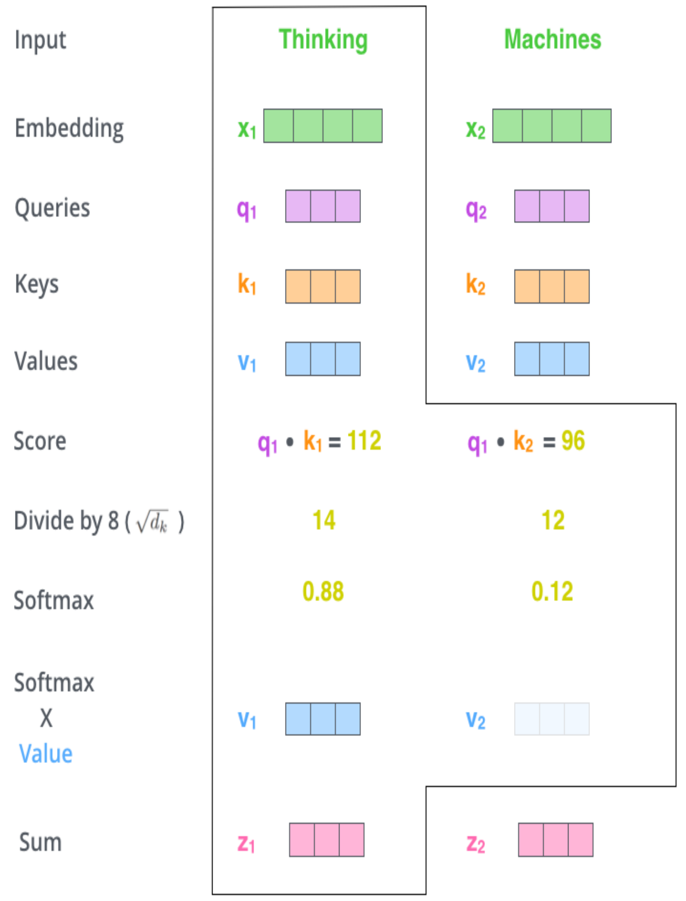
- Since our model contains no recurrence (and no convolution), in order for the model to make use of the order of the sequence, we add "positional encoding" to the input embedding
- The positional encodings have the same dimension d_{model} as the embeddings



$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Scaled Dot-Product Attention

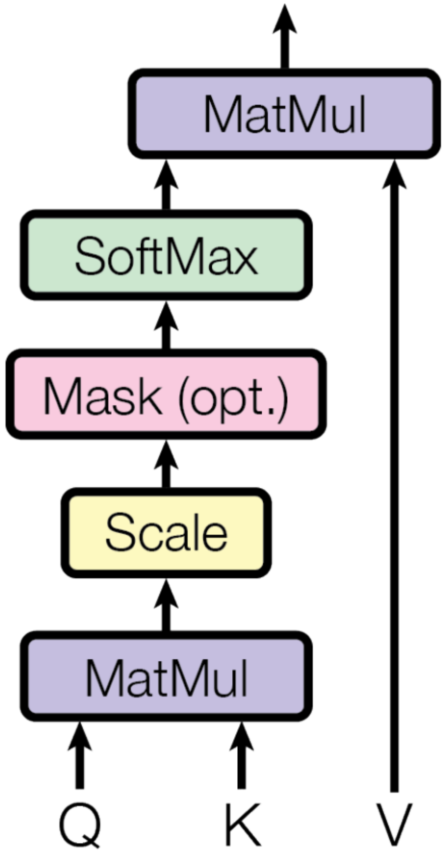




Scaled Dot-Product Attention

- We call our particular attention "Scaled Dot-Product Attention"
- The input consists of queries and keys of dimension d_k, and values of dimension d_v.
- We compute the dot products of the query with all keys, divide each by sqrt(d_k), and apply a softmax function to obtain the weights on the values

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

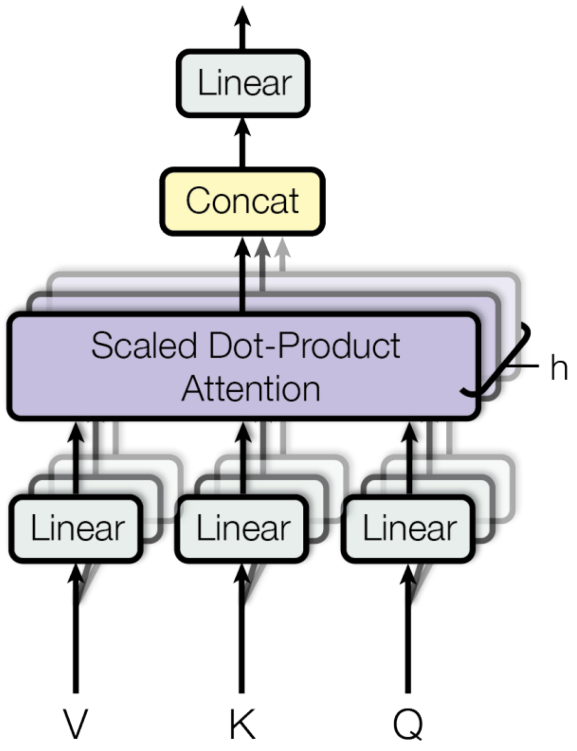


Multi-head Attention

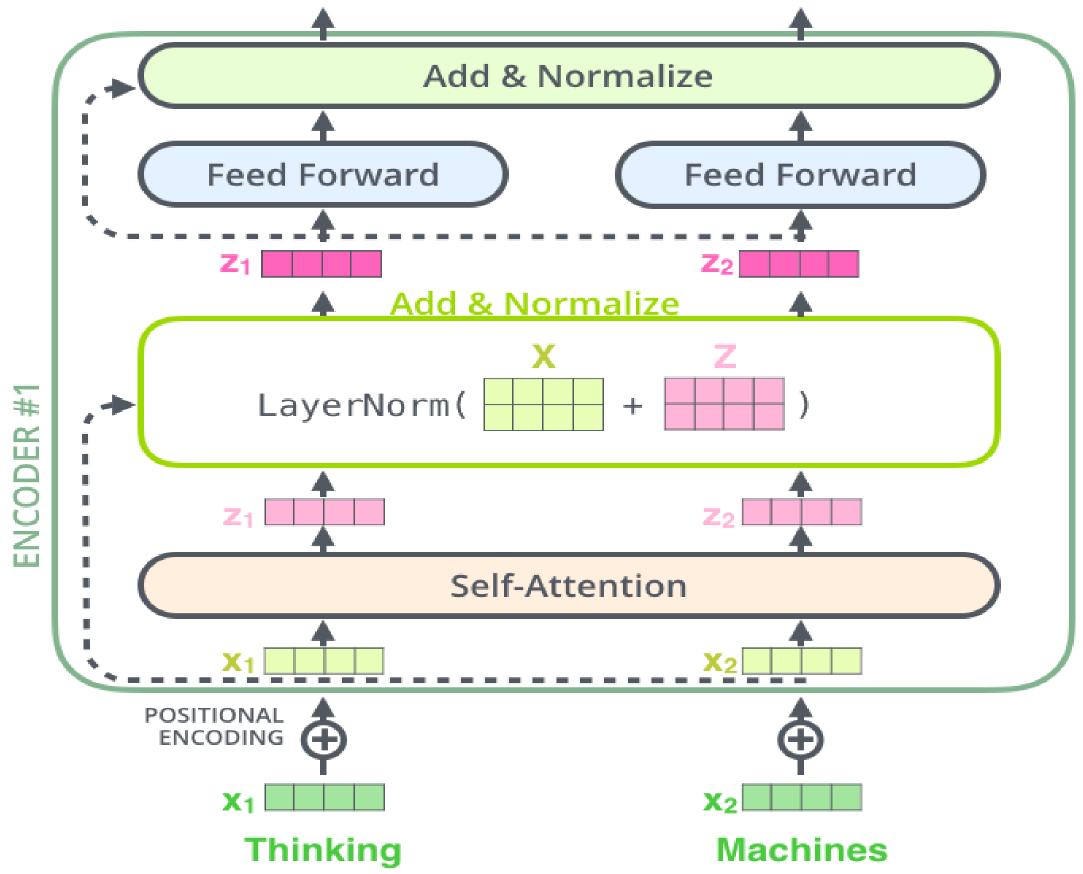
- Instead of performing a single attention, we found it beneficial to perform multi-head attention
- Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head}_1, ..., \mathrm{head}_\mathrm{h})W^O$$
$$\text{where } \mathrm{head}_\mathrm{i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Multi-Head Attention



Add & Norm


Add & Normalize
Feed Forward
Feed Forward
z_1
z_2
Add & Normalize
X
Z
LayerNorm(+)
z_1
z_2
Self-Attention
x_1
x_2
ENCODER #1
POSITIONAL ENCODING
x_1
x_2
Thinking
Machines

Feed Forward

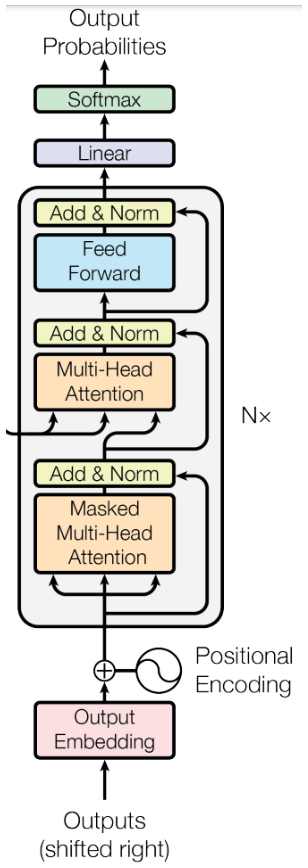
- Each of the layers in our encoder and decoder contains a fully connected feed-forward network. This consists of two linear transformations with a ReLU activation in between

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

- Add & Norm

3.2 Decoder Conclusion

- The decoder is also composed of a stack of N = 6 identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of encoder stack
- Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization
- We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i



Outputs (Shifted right) - Decoder Input

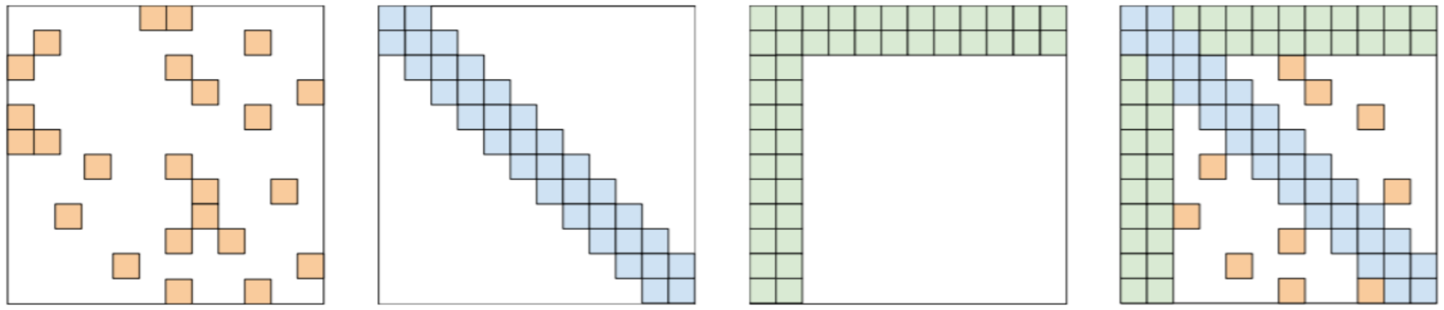
- Decoder 1: <s>
- Decoder 2: <s> I
- Decoder 3: <s> I am
- Decoder 4: <s> I am a
- Decoder 5: <s> I am a student --- STOP
- Decoder 6:

Third Sub-layer Input

- Output of the top encoder: z_1 z_2 z_3
- z_1 z_2 z_3 mul $W^K = k_1$ k_2 k_3
- z_1 z_2 z_3 mul $W^V = v_1$ v_2 v_3
- Output of mask multi-head attention: zz_1 zz_2
- zz_1 zz_2 mul $W^Q = q_1$ q_2
- Self-attention with q k v
- Multi-head attention

4. Drawbacks and Variants of Transformer

- Quadratic compute in self-attention
  - Computing all pairs of interactions means our computation grows quadratically with the sequence length
  - For recurrent models, it only grew linearly
- Key idea: replace all-pairs interactions with a family of other interactions, like local windows, looking at everything, and random interactions



(a) Random attention (b) Window attention (c) Global Attention (d) BIGBIRD

THANK YOU

Any questions?