Attention Mechanism Part 2

Zhengyang Wang

Some slides are adapted from Stanford CS224N/Ling284.
(Abigail See and Richard Socher)

Outlines

- Review the attention mechanism and self-attention
- Transformer: attention is all you need
- Non-local neural networks

Outlines

- Review the attention mechanism and self-attention
- Transformer: attention is all you need
- Non-local neural networks

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

$\boldsymbol{n_v = n_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boxed{\boldsymbol{d_q = d_k}}$

$\boldsymbol{n_v = n_k}$

"Soft" constraint:
Caused by dot product.

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

$\boxed{\boldsymbol{n_v = n_k}}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

"hard" constraint:
Caused by weighted sum.

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

$\boldsymbol{n_v = n_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boxed{\boldsymbol{d_v \times n_q}}}$$

  - The shape of the output is determined by the number of query vectors and the dimension of value vectors.

Self-Attention

- Use X (itself) to generate Q, K, V and perform attention.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$ if using dot product to compute attention scores

- What is the shape of the output Y ?

$$\boldsymbol{d_v \times n_x}$$

- **If we omit $\boldsymbol{W_v}$ and simply have $\boldsymbol{V = X}$, each column in $\boldsymbol{Y}$ is a weighted sum of all column vectors in $\boldsymbol{X}$.**

Self-Attention

- **If we omit W_v and simply have $V = X$, each column in Y is a weighted sum of all column vectors in X.**

- Comparison with convolution and fully-connected layer
  - Convolution
    - Each column in Y is computed from column vectors within a local range in X.
  - Fully-connected layer
    - The weights in the weighted sum is not input-dependent.

Multi-Head Self-Attention

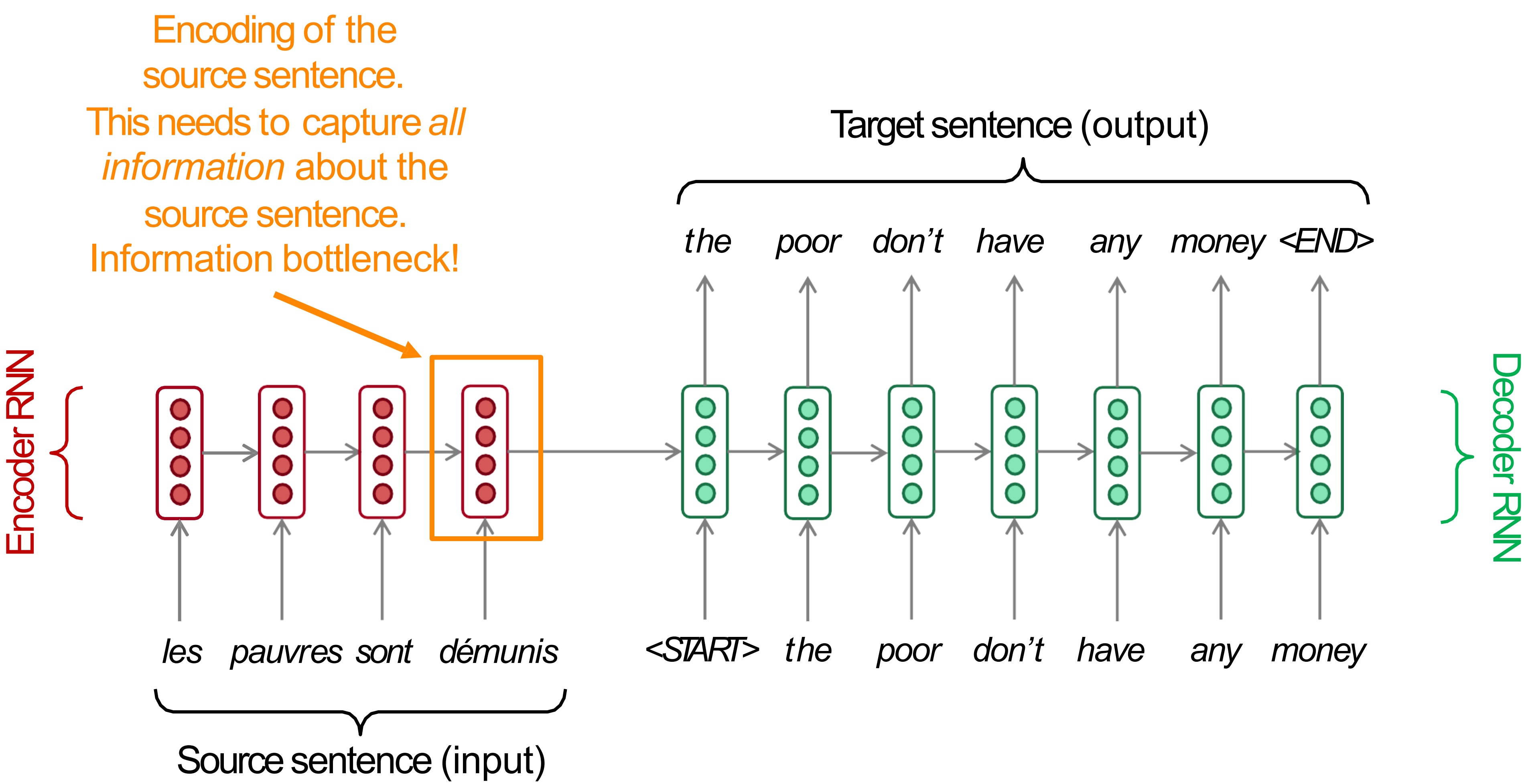
- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

- Do the above process for multiple times <u>independently</u>.
  - Use $W_q^{(i)}, W_k^{(i)}, W_v^{(i)}$ to get Q_i, K_i, V_i and do attention <u>independently.</u>
  - Each group results in an output Y_i.
  - Concatenate all the Y_i and go through a linear transformation to obtain the final output.

Outlines

- Review the attention mechanism and self-attention
- Transformer: attention is all you need
- Non-local neural networks

Sequence-to-sequence: the bottleneck problem

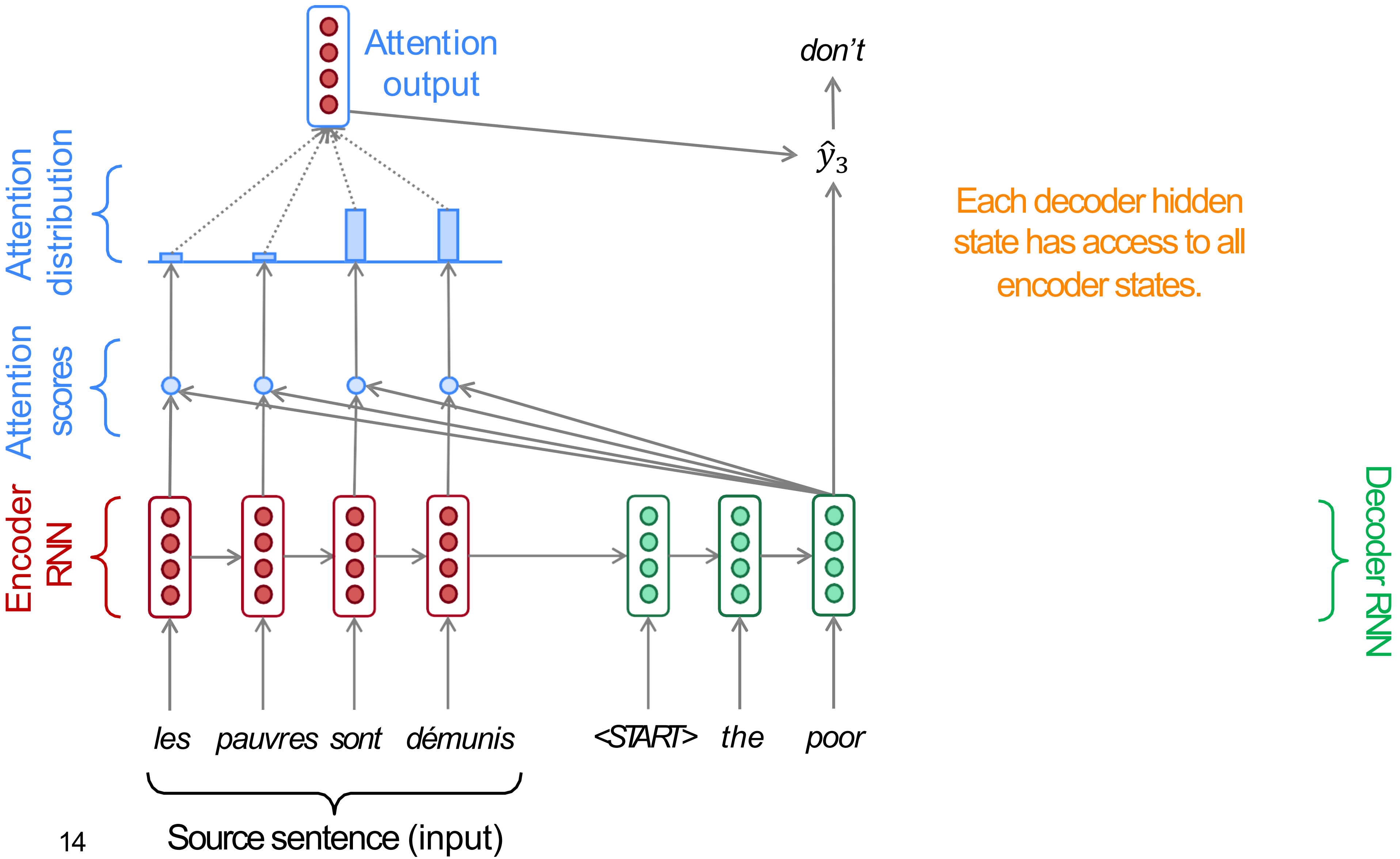


Attention

- **Attention** provides a solution to the bottleneck problem.

- Core idea: on each step of the decoder, *focus on a particular part* of the source sequence

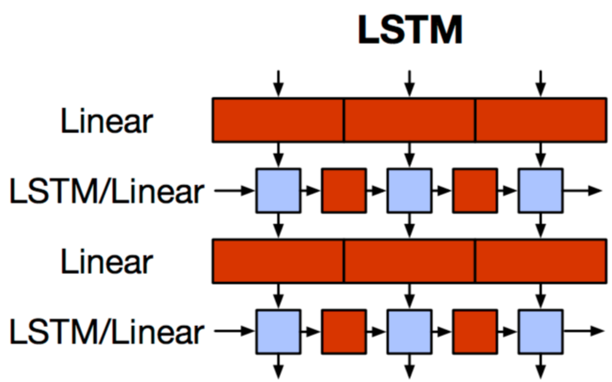
Sequence-to-sequence with attention



Each decoder hidden state has access to all encoder states.

Problems with RNNs = Motivation for Transformers

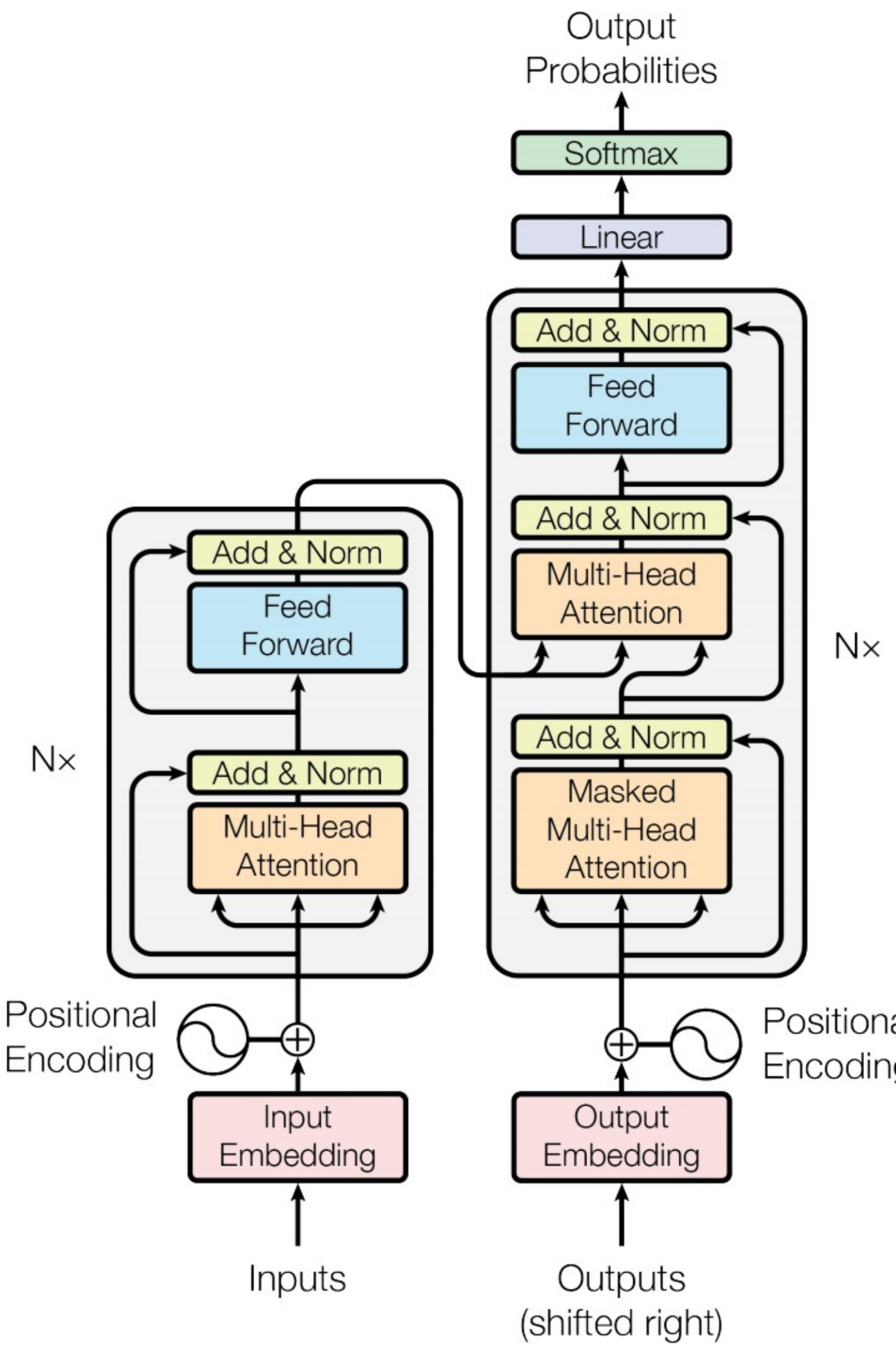
- Sequential computation prevents parallelization



- Despite GRUs and LSTMs, RNNs still need attention mechanism to deal with long range dependencies – path length for co-dependent computation between states grows with sequence
- But if attention gives us access to any state… maybe we don't need the RNN? 🤔

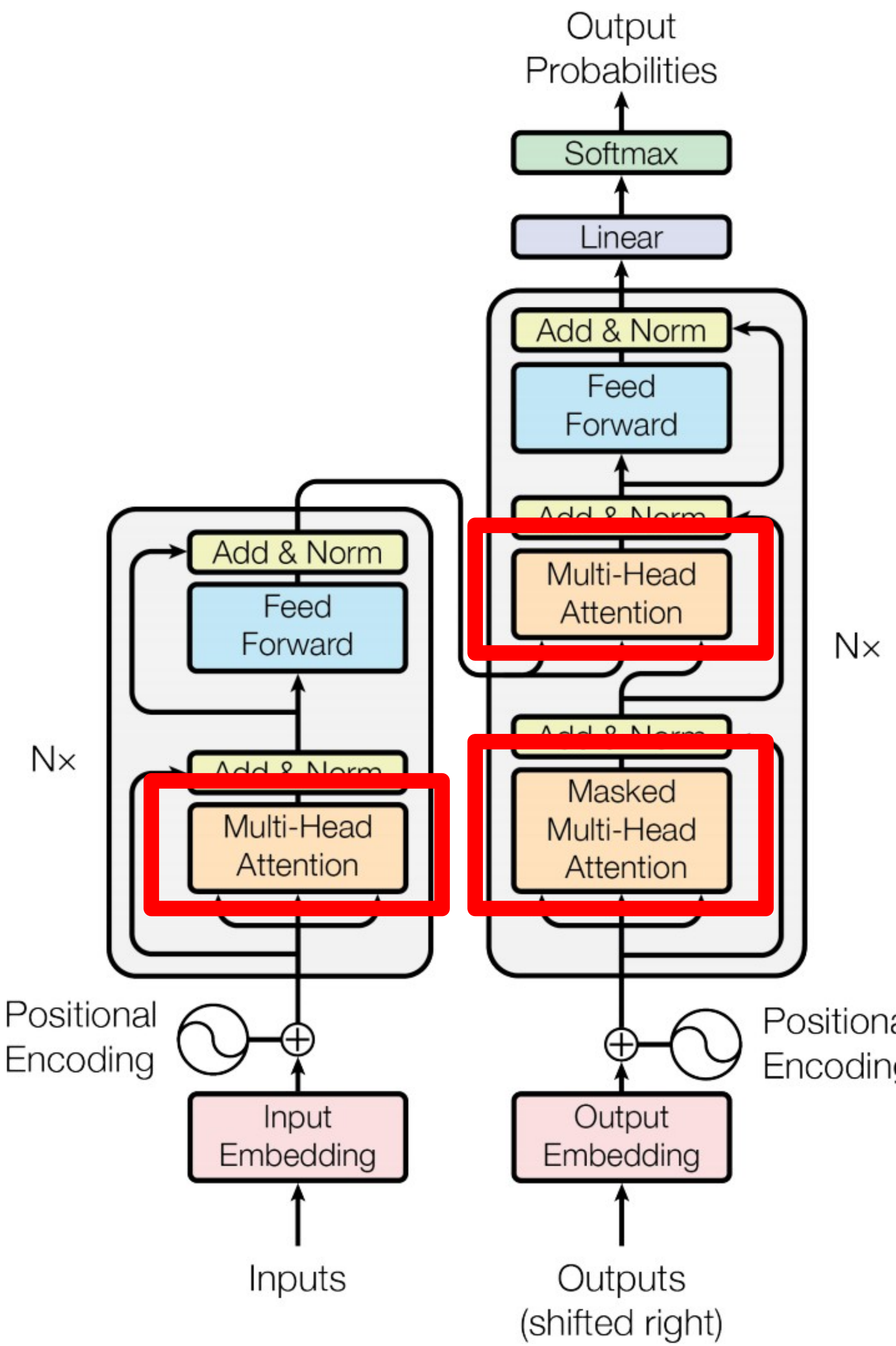
Transformer Overview

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



Transformer Overview

- Sequence-to-sequence
- Encoder-Decoder
- Task: machine translation with parallel corpus
- Predict each translated word
- Final cost/error function is standard cross-entropy error on top of a softmax classifier



Transformer Basics

- Let's define the basic building blocks of transformer networks first: new attention layers!

- Just the same attention, with different settings.

Ours:

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$$\boldsymbol{d_q = d_k}$$

$$\boldsymbol{n_v = n_k}$$

Transformer:

- Three input matrices:
  - $Q \in \mathbb{R}^{|Q| \times d_k}$
  - $K \in \mathbb{R}^{|K| \times d_k}$
  - $V \in \mathbb{R}^{|K| \times d_v}$

Dot-Product Attention (Extending our previous def.)

- Inputs: a query q and a set of key-value (k-v) pairs to an output
- Query, keys, values, and output are all vectors

- Output is weighted sum of values, where
- Weight of each value is computed by an inner product of query and corresponding key
- Queries and keys have same dimensionality d_k value have d_v

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

Dot-Product Attention - Matrix notation

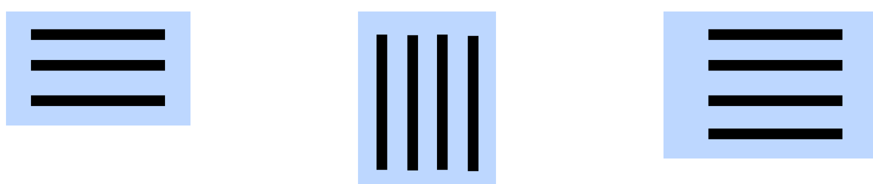
- When we have multiple queries q, we stack them in a matrix Q:

$$A(q, K, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

- Becomes: $A(Q, K, V) = softmax(QK^T)V$

$[|Q| \times d_k] \times [d_k \times |K|] \times [|K| \times d_v]$

softmax
row-wise



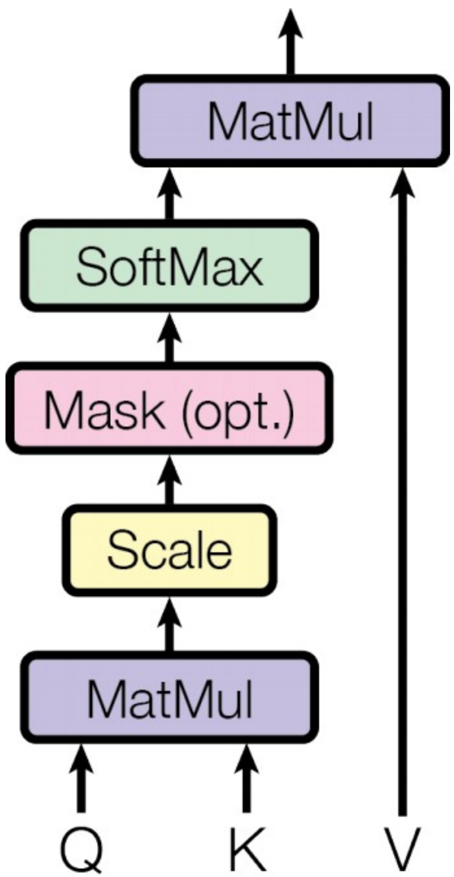
$= [|Q| \times d_v]$

Scaled Dot-Product Attention

- Problem: As d_k gets large, the variance of q^Tk increases → some values inside the softmax get large → the softmax gets very peaked --> hence its gradient gets smaller.

- Solution: Scale by length of query/key vectors:

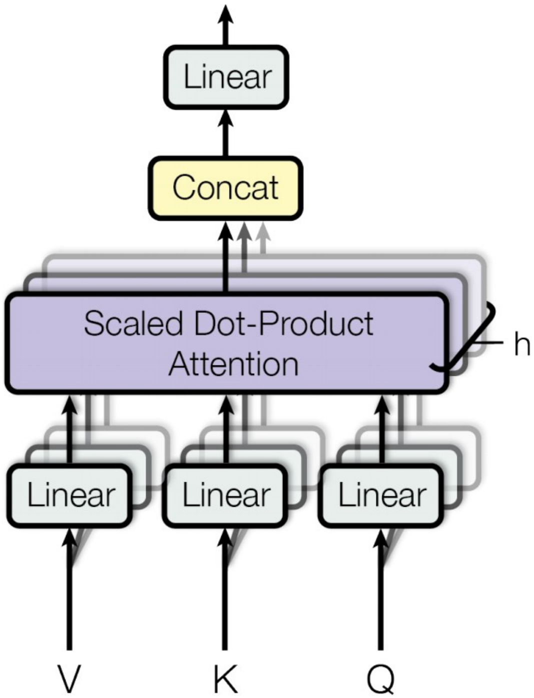
$$A(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$



Self-attention and Multi-head attention

- The input word vectors could be the queries, keys and values
- In other words: the word vectors themselves select each other
- Word vector stack = Q = K = V
- Problem: Only one way for words to interact with one-another
- Solution: Multi-head attention
- First map Q, K, V into h many lower dimensional spaces via W matrices
- Then apply attention, then concatenate outputs and pipe through linear layer

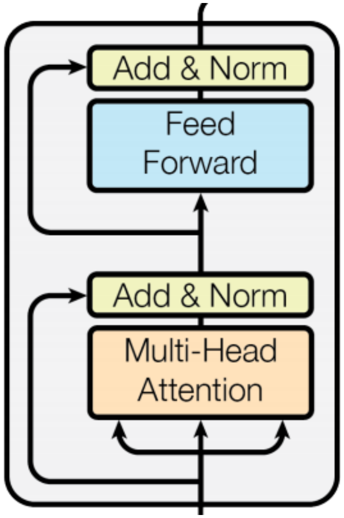
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Complete transformer block

- Each block has two "sublayers"

1. Multihead attention
2. 2 layer feed-forward Nnet (with relu)



Each of these two steps also has:
Residual (short-circuit) connection and LayerNorm:

LayerNorm(x + Sublayer(x))

Layernorm changes input to have mean 0 and variance 1,
per layer and per training point (and adds two more parameters)

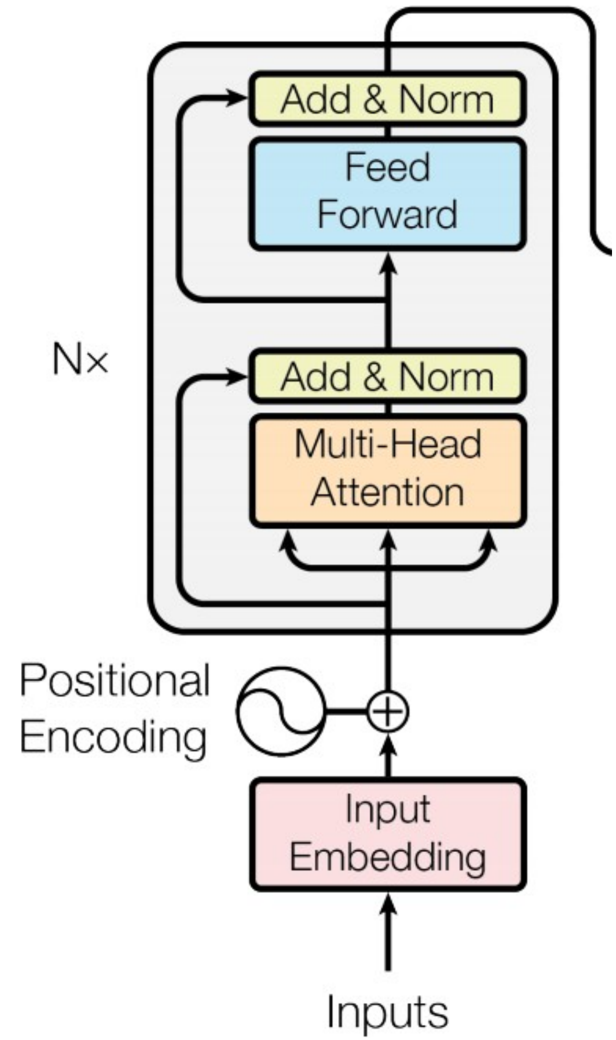
$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad h_i = f\left(\frac{g_i}{\sigma_i}\left(a_i - \mu_i\right) + b_i\right)$$

Layer Normalization by Ba, Kiros and Hinton, https://arxiv.org/pdf/1607.06450.pdf

Complete Encoder

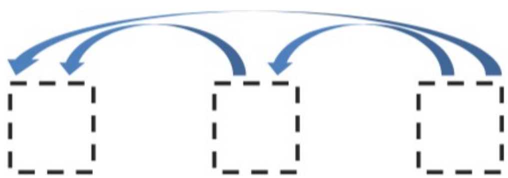
- For encoder, at each block, we get the Q, K and V from the output of the previous layer.

- Blocks are repeated 6 times.

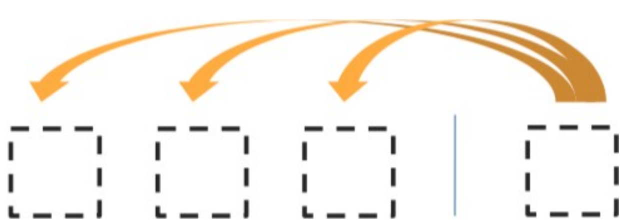


Transformer Decoder

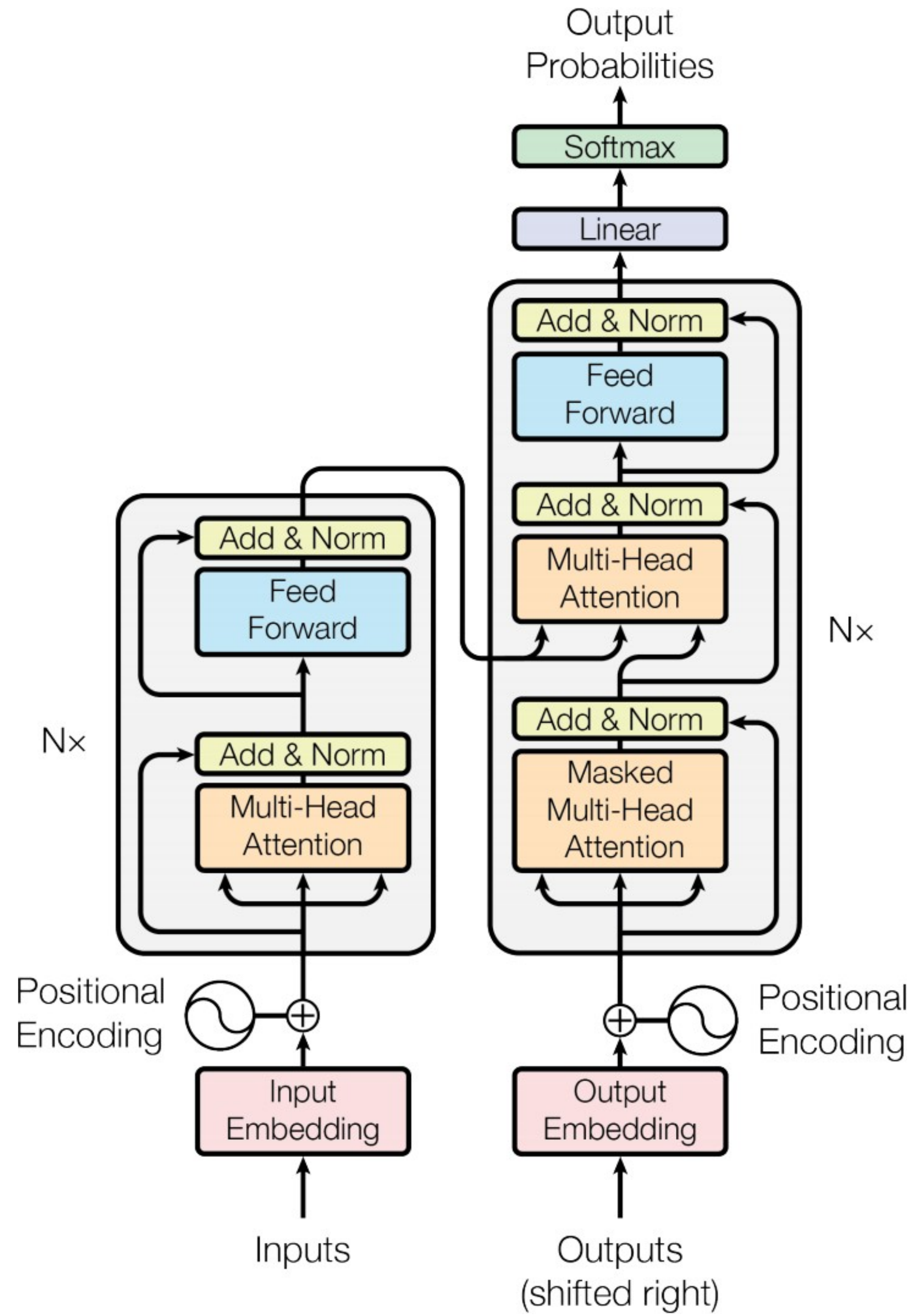
- 2 sublayer changes in decoder
- Masked decoder self-attention on previously generated outputs:



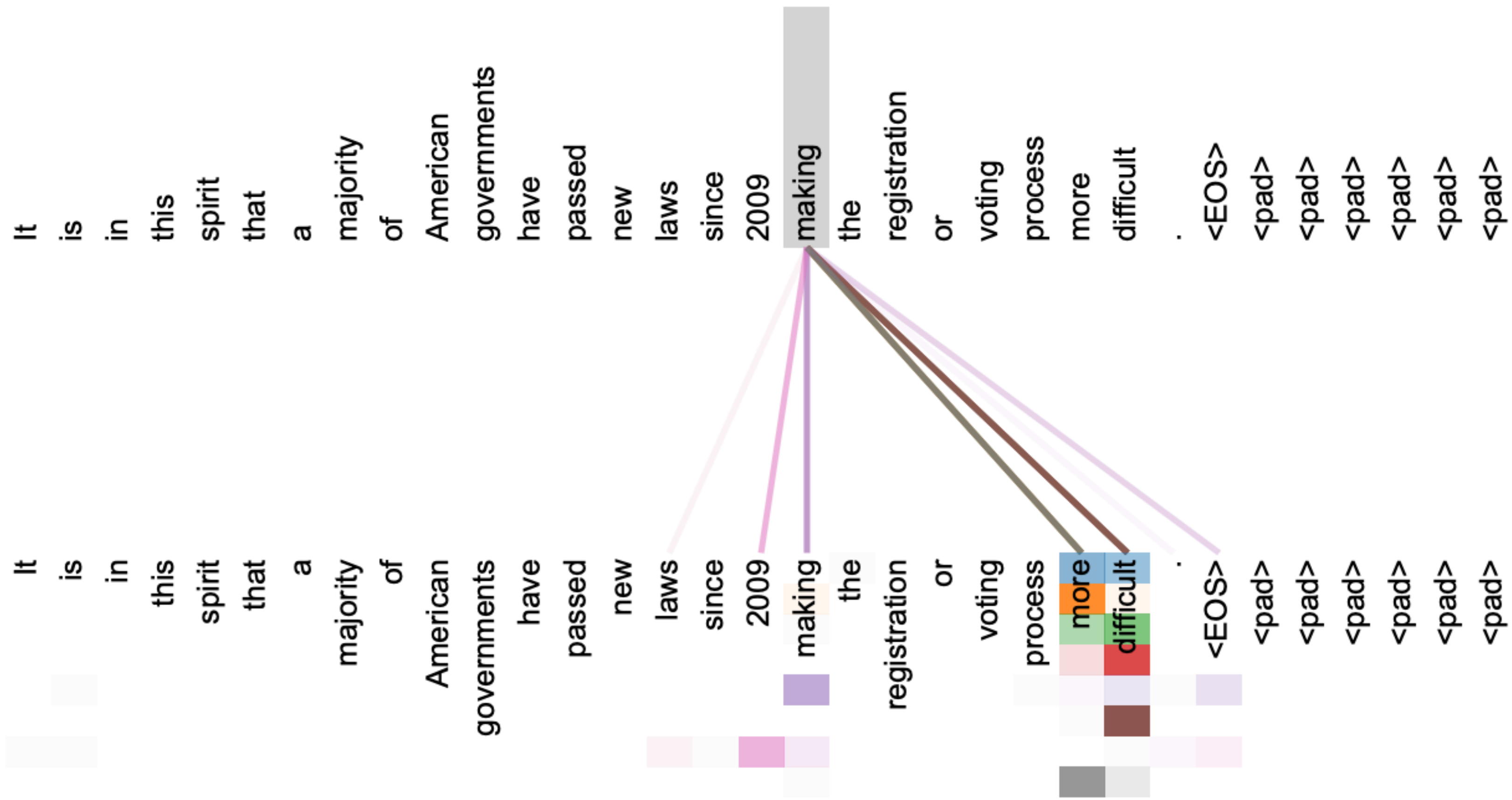
- Encoder-Decoder Attention, where queries come from previous decoder layer and keys and values come from output of encoder

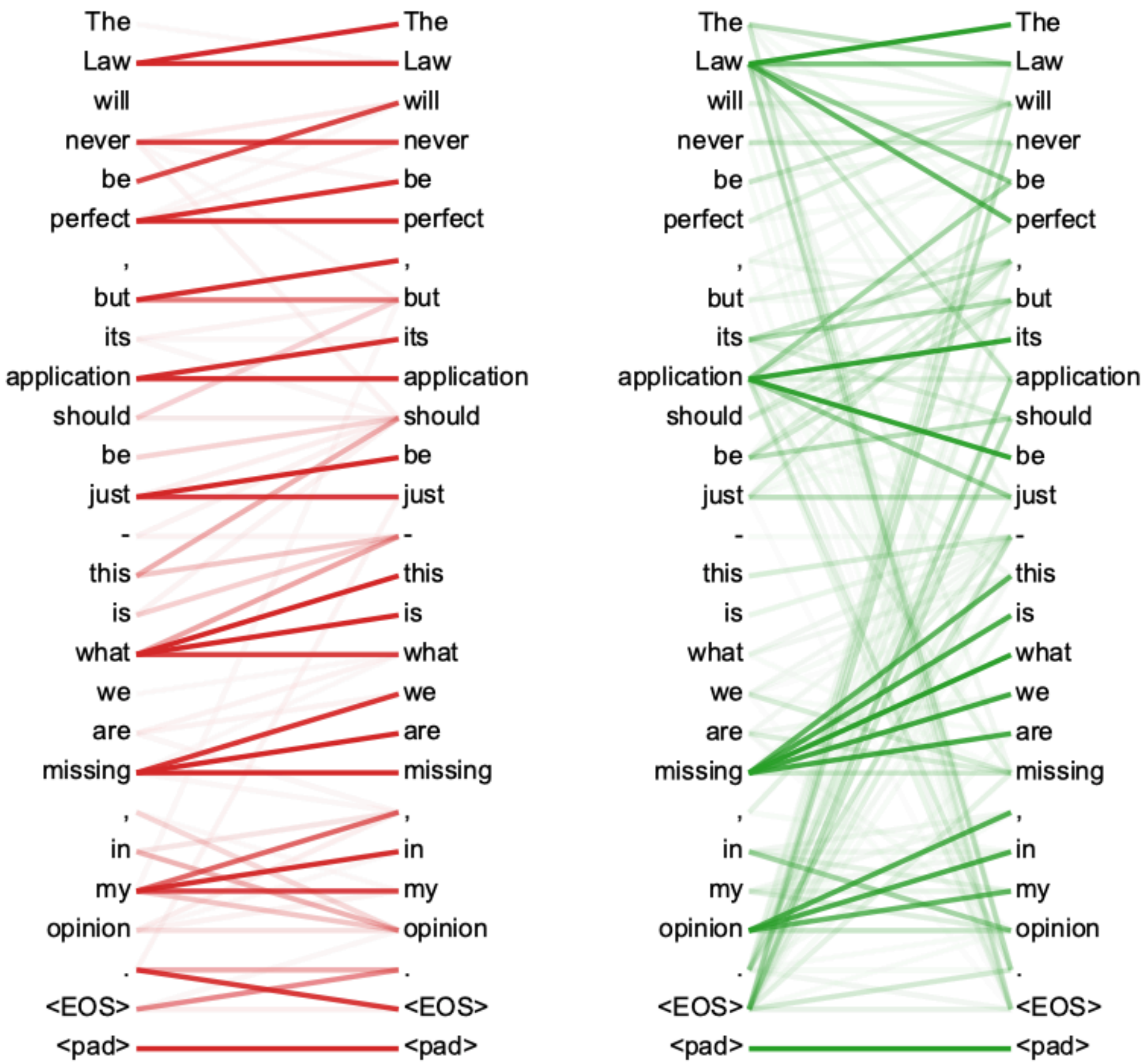


- Repeat 6 times also



Attention Visualizations




The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>

Outlines

- Review the attention mechanism and self-attention
- Transformer: attention is all you need
- Non-local neural networks

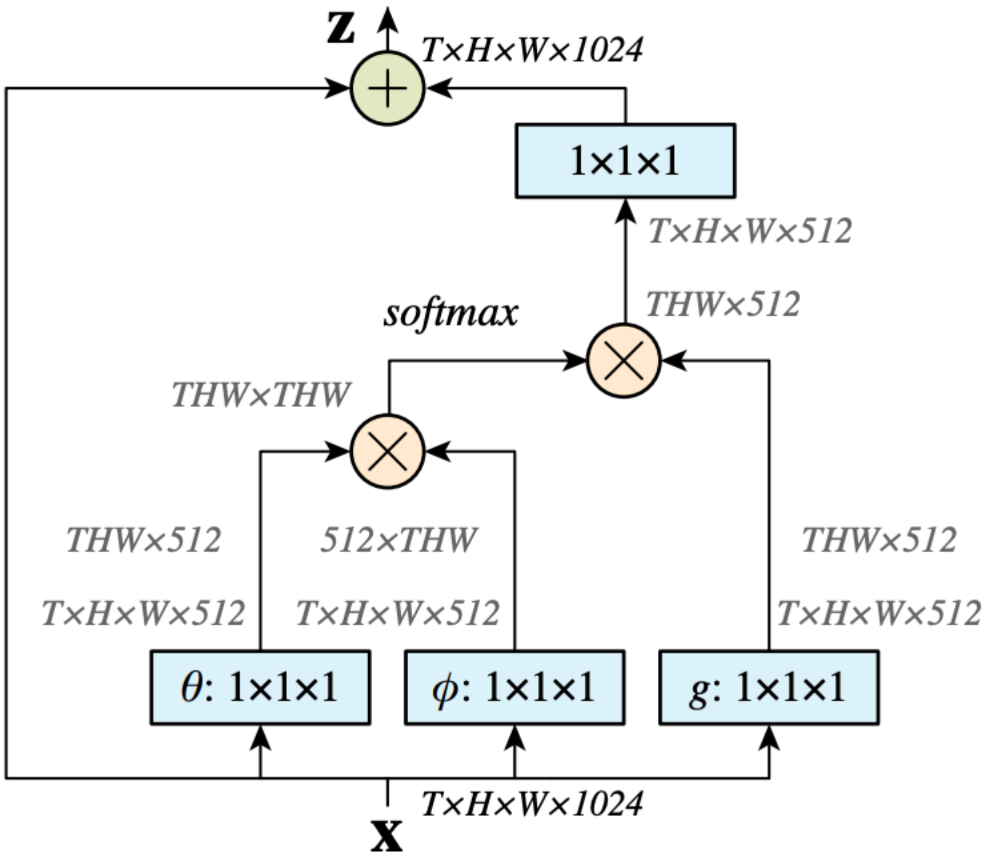
Non-local Neural Networks

- High-dimensional case of using self-attention
  - Texts are 1-D data
  - Images are 2-D data
  - Videos are 3-D data
  - …

- Combined usage of self-attention and convolution
  - Convolution: extract local features
  - Self-attention: aggregate non-local information

- The paper discussed different ways of computing and normalizing attention scores.

Non-local Neural Networks



Non-local Neural Networks

layer		output size
conv_1	7×7, 64, stride 2, 2, 2	$16\times112\times112$
pool_1	$3\times3\times3$ max, stride 2, 2, 2	$8\times56\times56$
res_2	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$8\times56\times56$
pool_2	$3\times1\times1$ max, stride 2, 1, 1	$4\times56\times56$
res_3	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$4\times28\times28$
res_4	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$4\times14\times14$
res_5	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$4\times7\times7$
global average pool, fc		$1\times1\times1$

Insert the self-attention block here!