VIRGINIA TECH

Attention and Transformer

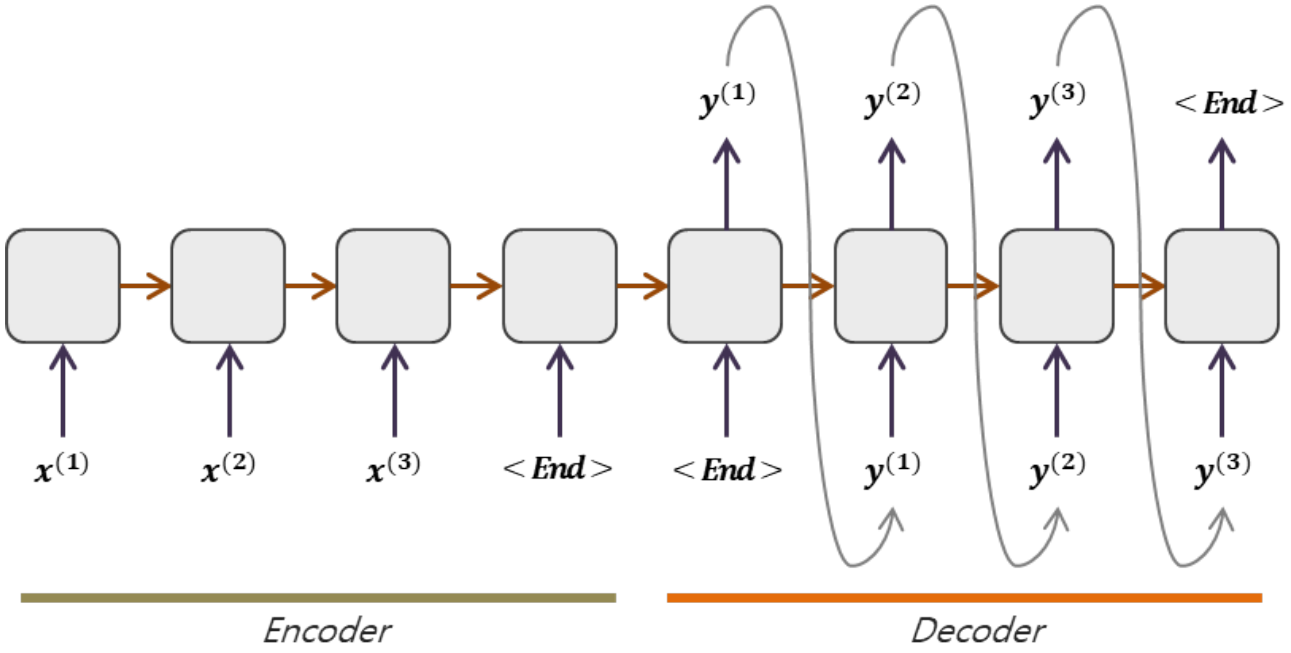
Lifu Huang

lifuh@vt.edu

Torgersen Hall, Suite 3160E

Sequence to Sequence (Seq2Seq) Models

- also known as Encoder-Decoder networks



Encoder: get a hidden state *h* for each input *x*

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

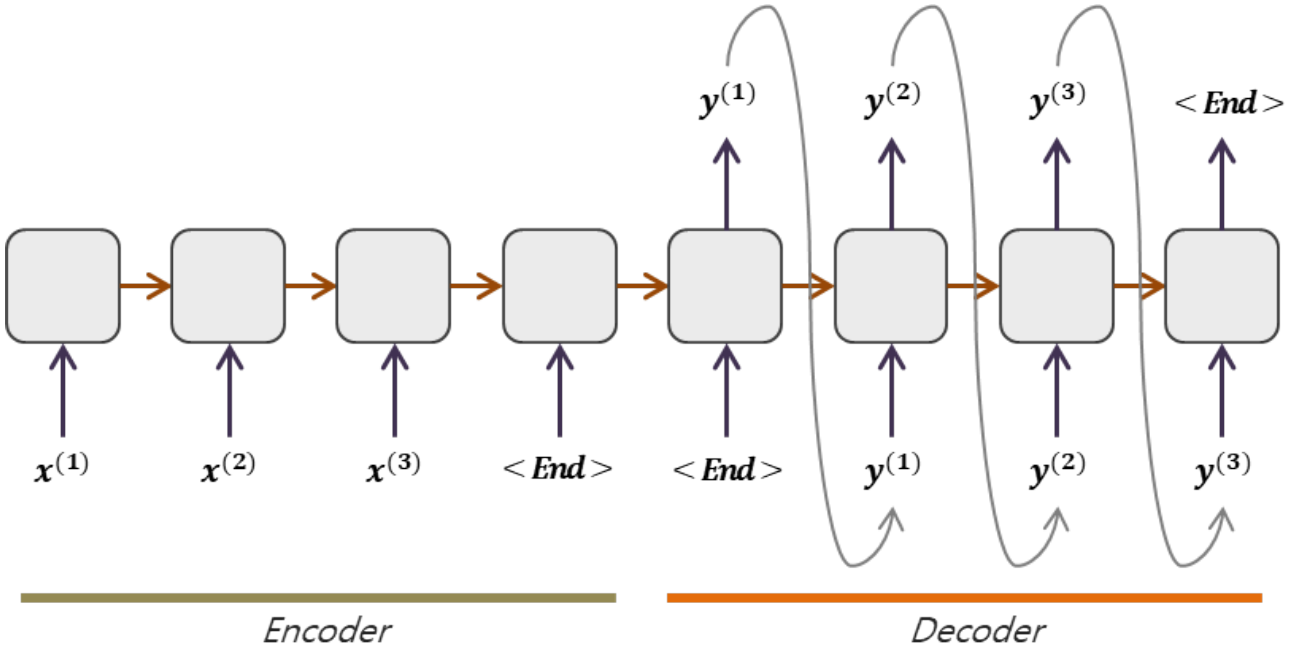
Decoder: predict an output *y* given a hidden state *h*

$$h_t = f(W^{(hh)} h_{t-1} + W^{hy} y_{t-1})$$

$$y_t = softmax(W^S h_t)$$

Sequence to Sequence (Seq2Seq) Models

- also known as Encoder-Decoder networks



Encoder: get a hidden state *h* for each input *x*

$$h_t = f(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

Decoder: predict an output *y* given a hidden state *h*

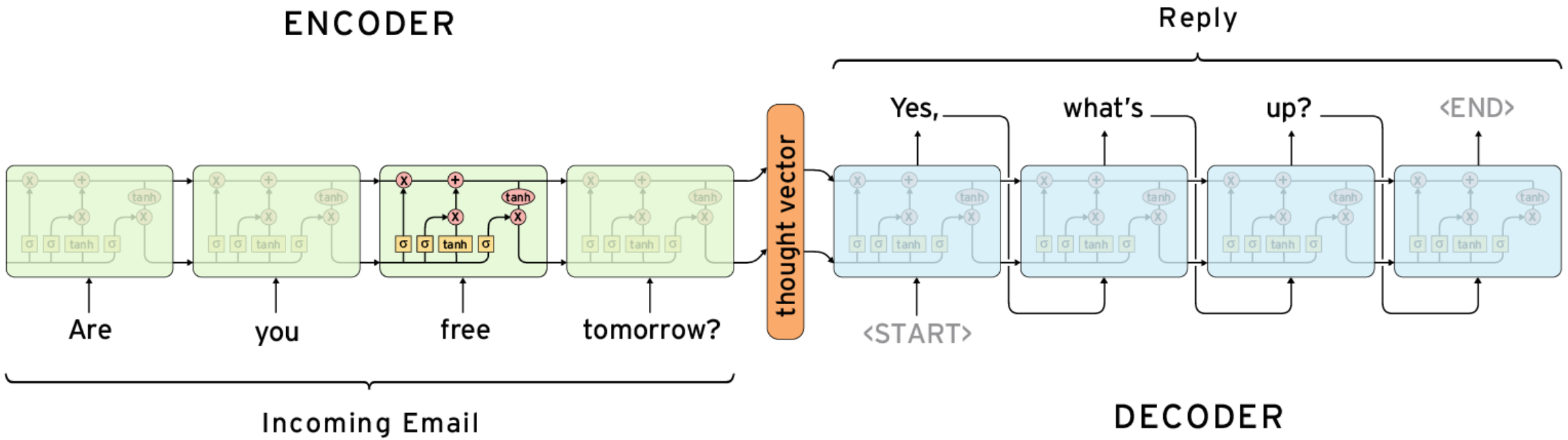
$$h_t = f(W^{(hh)} h_{t-1} + W^{hy} y_{t-1})$$

$$y_t = softmax(W^S h_t)$$

Problem: the performance of the seq2seq model degrades rapidly as the length of the input sentence increases.

Seq2Seq - Application

- Generating a response given a previous message with LSTM based Encoder-Decoder framework

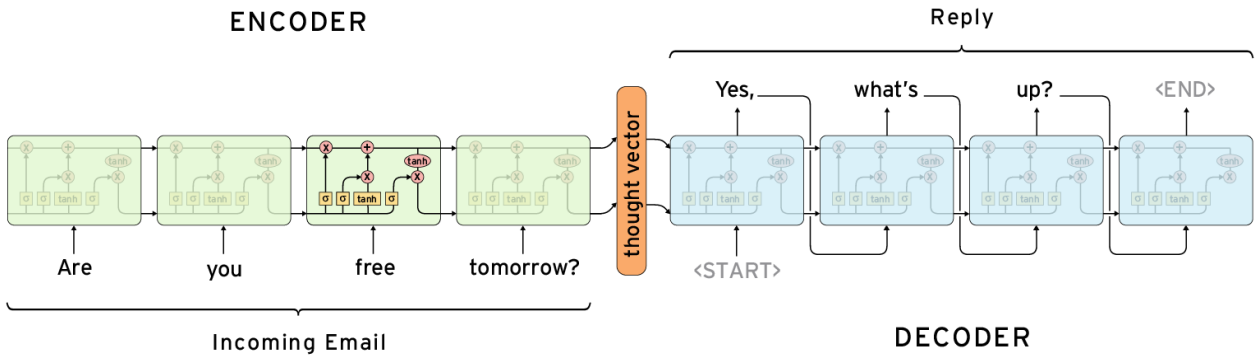


What is Attention

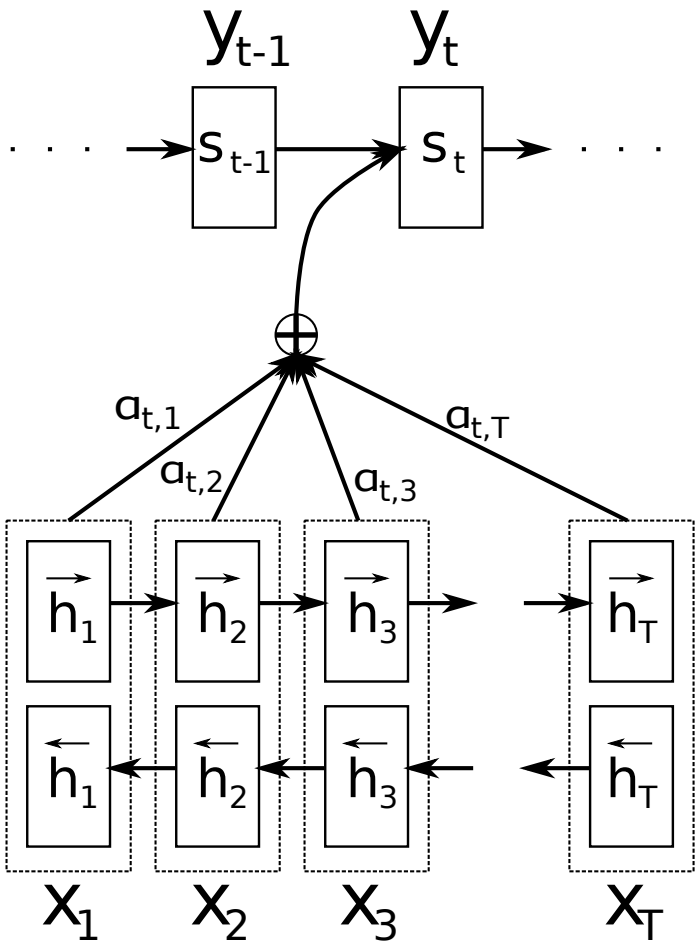
- Attention – Directing your (the machine's) focus at something and then perform the task, e.g., prediction

- In deep learning, it's used to manage and quantify the interdependence:
  - Between the input and output elements (generation attention)
  - Within the input elements (self-attention)
  - Only select a few input elements to focus on (hard attention)
  - Focus on all the input elements but assign soft weights to various locations (soft attention)

- Analogy …



Seq-to-Seq v.s. Seq-to-Seq with Attention



$$e_{ij} = a(s_{i-1}, h_j)$$ ← Attention Score

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$ ← Normalized Attention Weights

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$ ← Weighted sum of Encoder hidden states

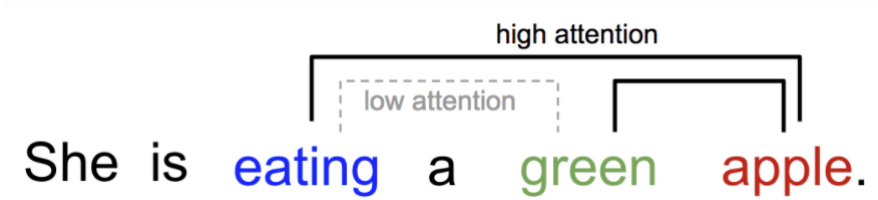
$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$ ← Compute the next Decoder hidden state

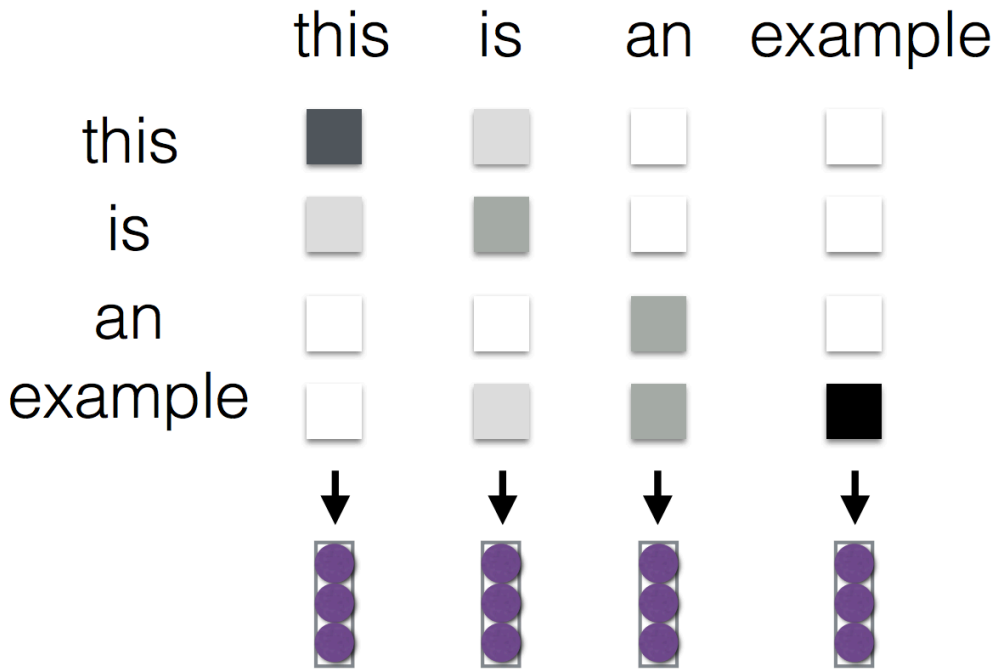
Attention Calculation

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Self-attention

- Each element in the sentence attends to other elements → context sensitive encodings!





Transformer – Attention is all you need, Vaswani et al. 2017

- A sequence-to-sequence model based entirely on attention
- Strong results on standard WMT datasets
- Fast: only matrix multiplications

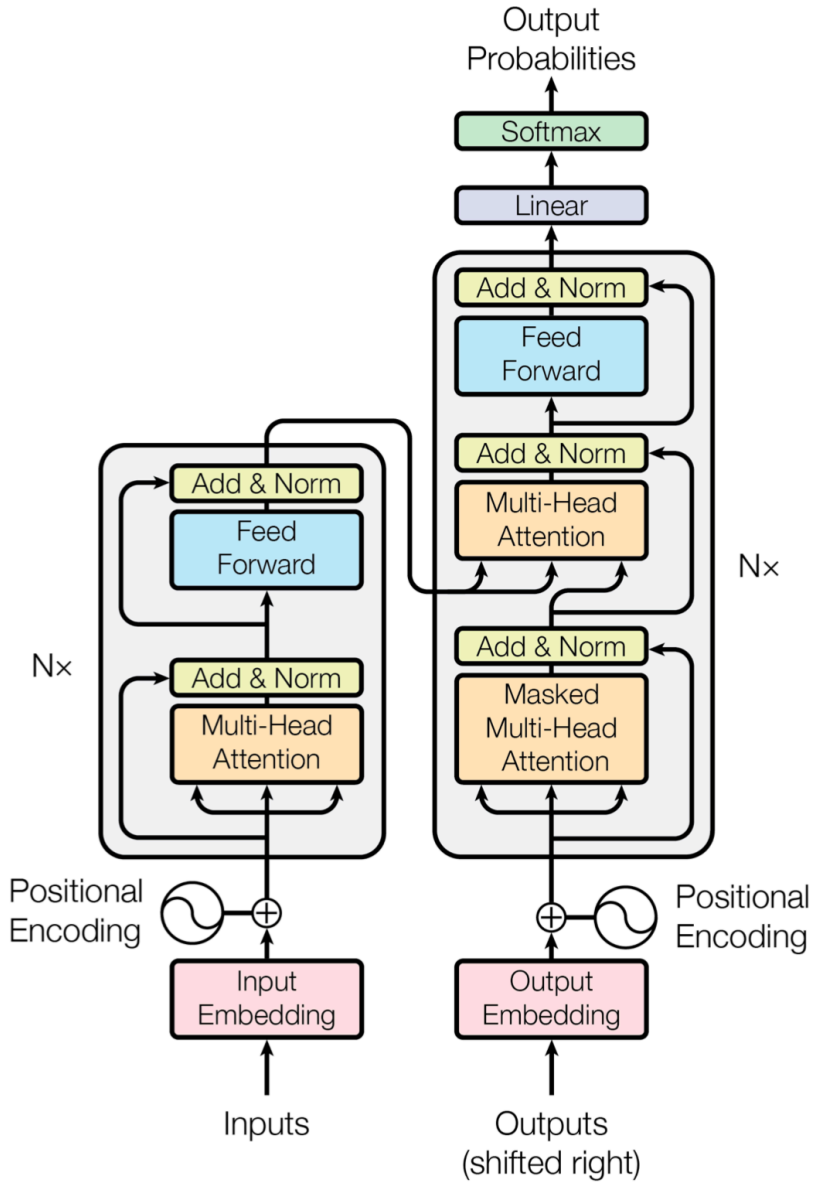
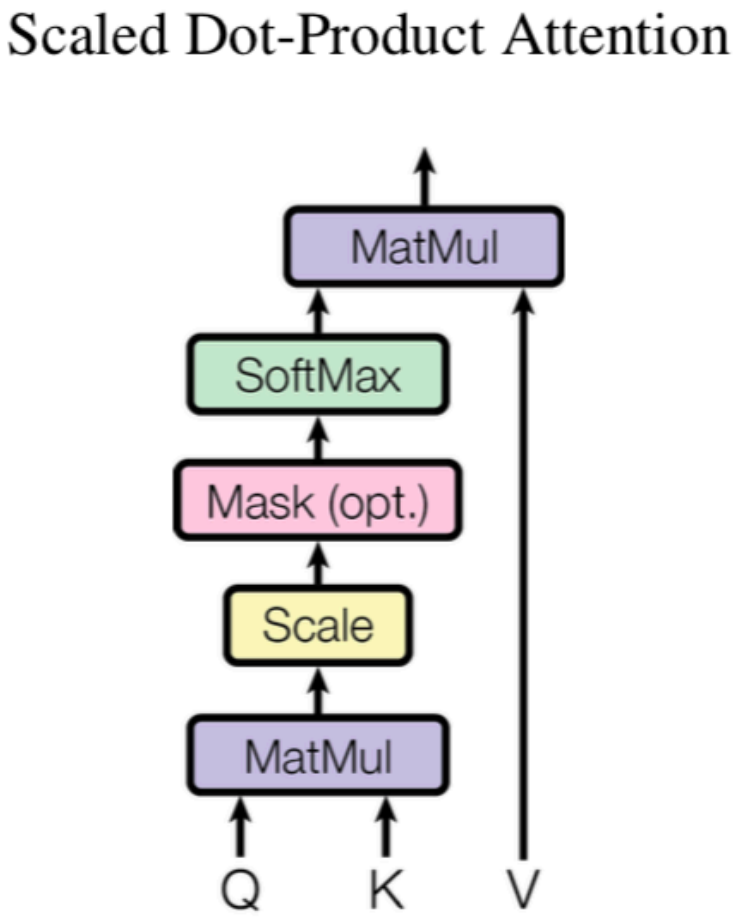


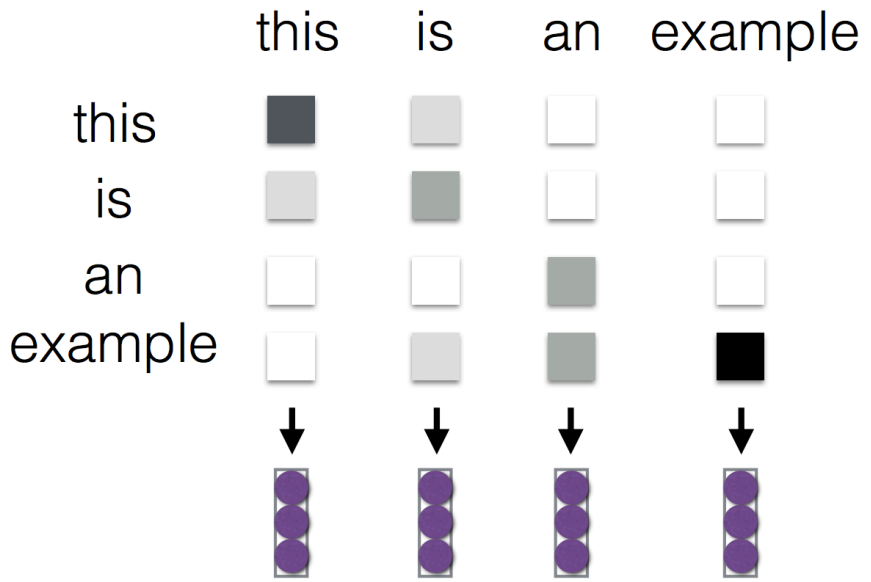
Figure 1: The Transformer - model architecture.

Scaled Dot-Product Self-Attention

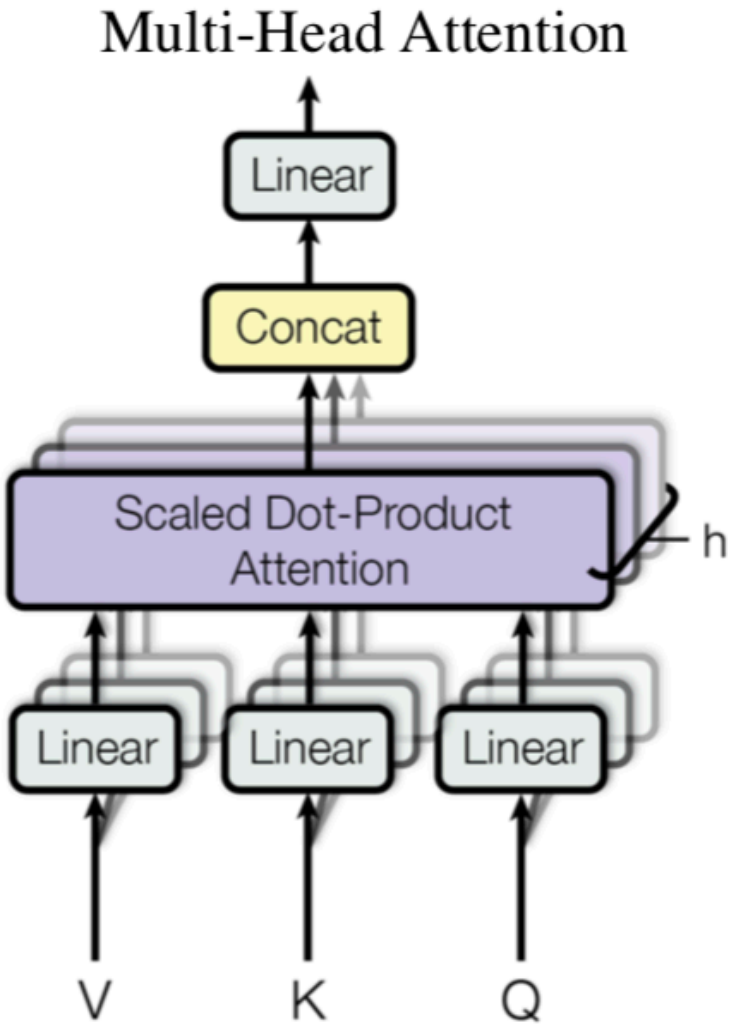


Attention: mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors

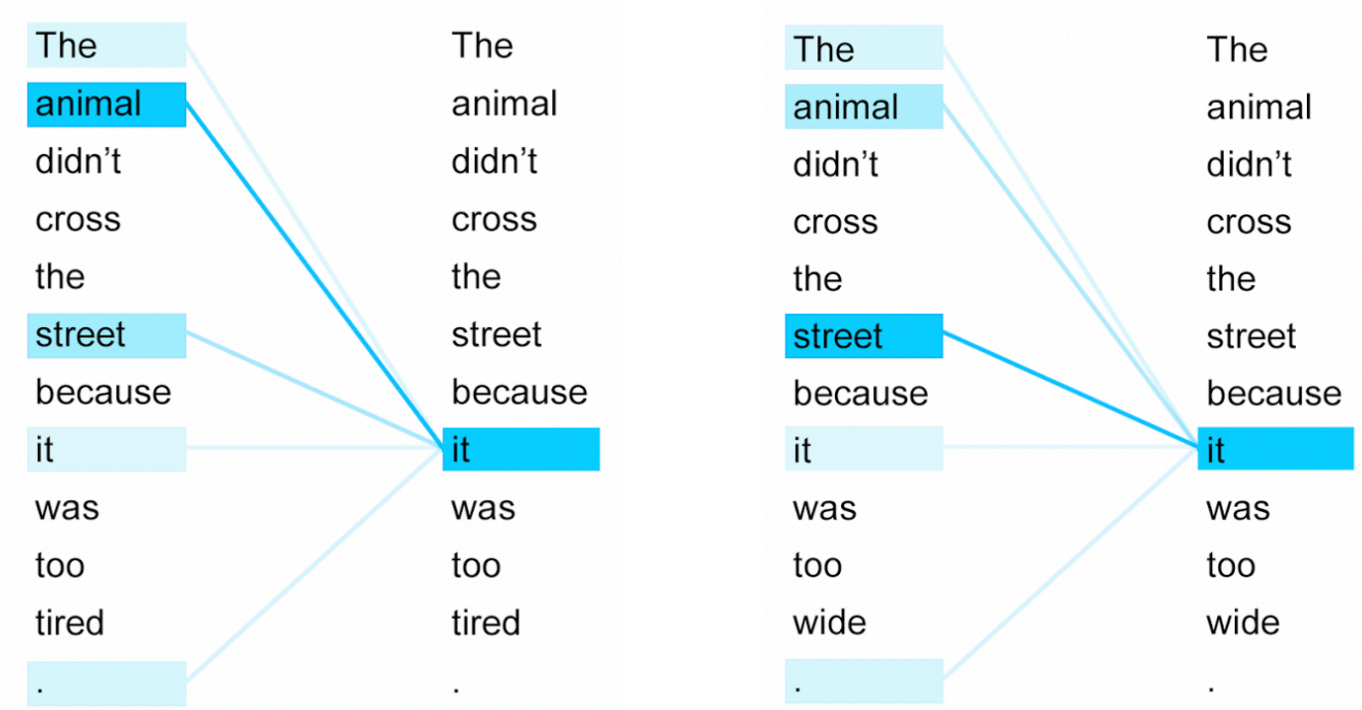
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Multi-head Self-Attention



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



How to capture the position or order of words in a sentence?

- Both RNNs and CNNs are aware of the positions of words in a sentence
- Positional Embedding

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

pos: position of the word
i: dimension

- Any other positional embeddings will work?

- Input Embedding = Word Embedding + Positional Embedding

How about the decoder?

- At the decoding time step *t*
  - Multi-head self-attention among all the previous *t-1* outputs
    - Mask out the future outputs
  - Multi-head (general) attention
    - Similar as the attention in seq2seq models
    - **Query** -> output of the decoder multi-head self-attention
    - **Key** -> output of the encoder multi-head self-attention
    - **Value** -> output of the encoder multi-head self-attention

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

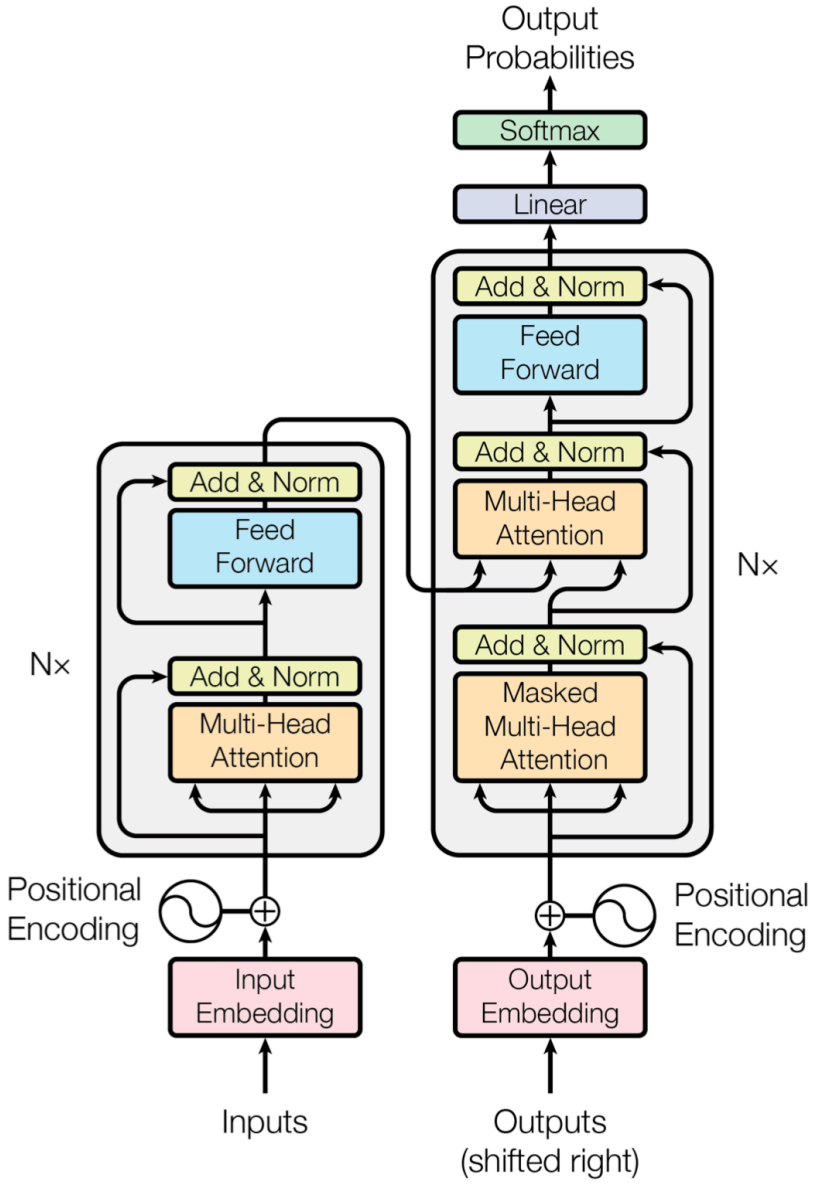


Figure 1: The Transformer - model architecture.

Overall Architecture

- Feedforward Network

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

ReLU Activation

- Add & Norm
  - Additive Residual Connection

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$

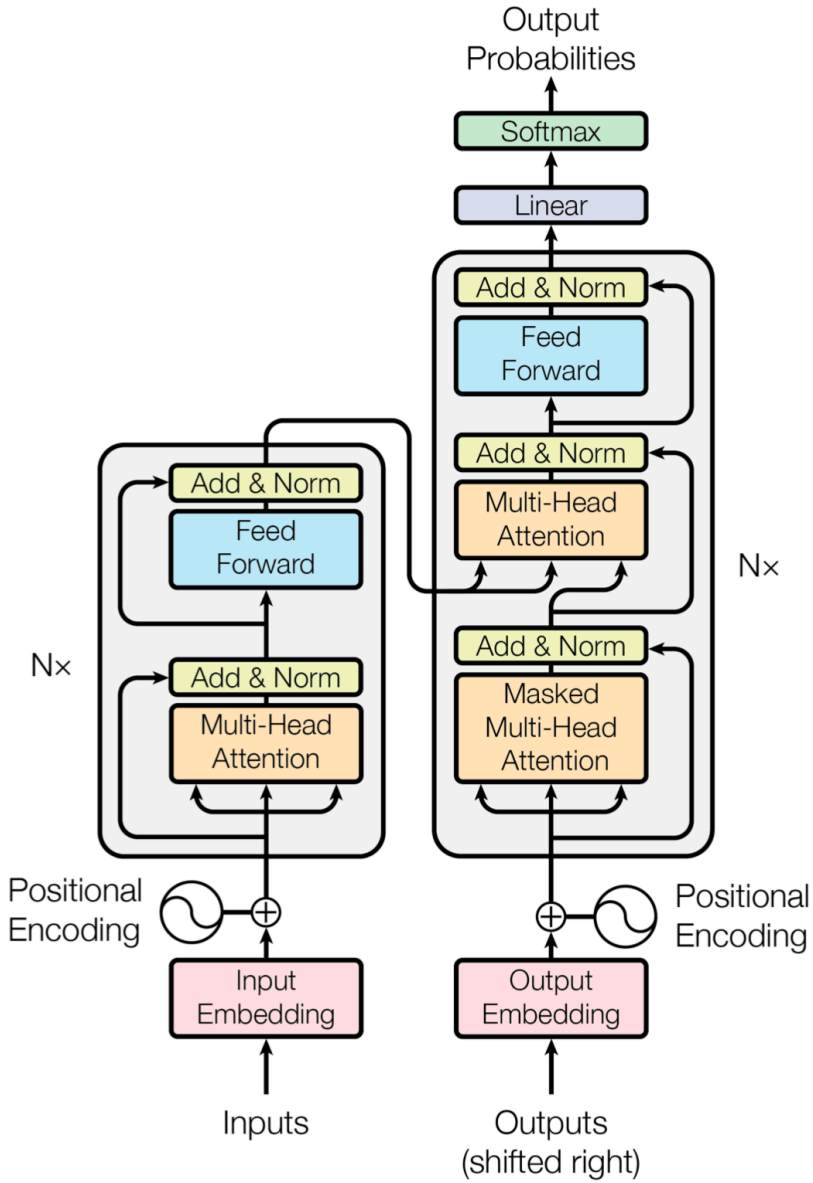


Figure 1: The Transformer - model architecture.

Training Tricks

- **Layer Normalization**
  - Help ensure that layers remain in reasonable range and make the model converge faster
- **Specialized Training Schedule**
  - Adjust default learning rate of the Adam optimizer
- **Label Smoothing**
  - Insert some uncertainty in the training process

$$y_{ls} = (1 - \epsilon_{ls}) \cdot y_{hot} + \epsilon_{ls}/K$$

Coefficient — One-hot label vector — Number of classes

Prevent overconfident predictions

Evaluation of Transformer on Machine Translation

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	positional embedding instead of sinusoids									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

N : number of layers

d_{model} : dimension size of key/ value/query vectors

d_{ff} : hidden dimension size of feed-forward network

h : number of heads

d_k : projected dimension size
d_v of key/value/query vectors

P_{drop} : dropout rate

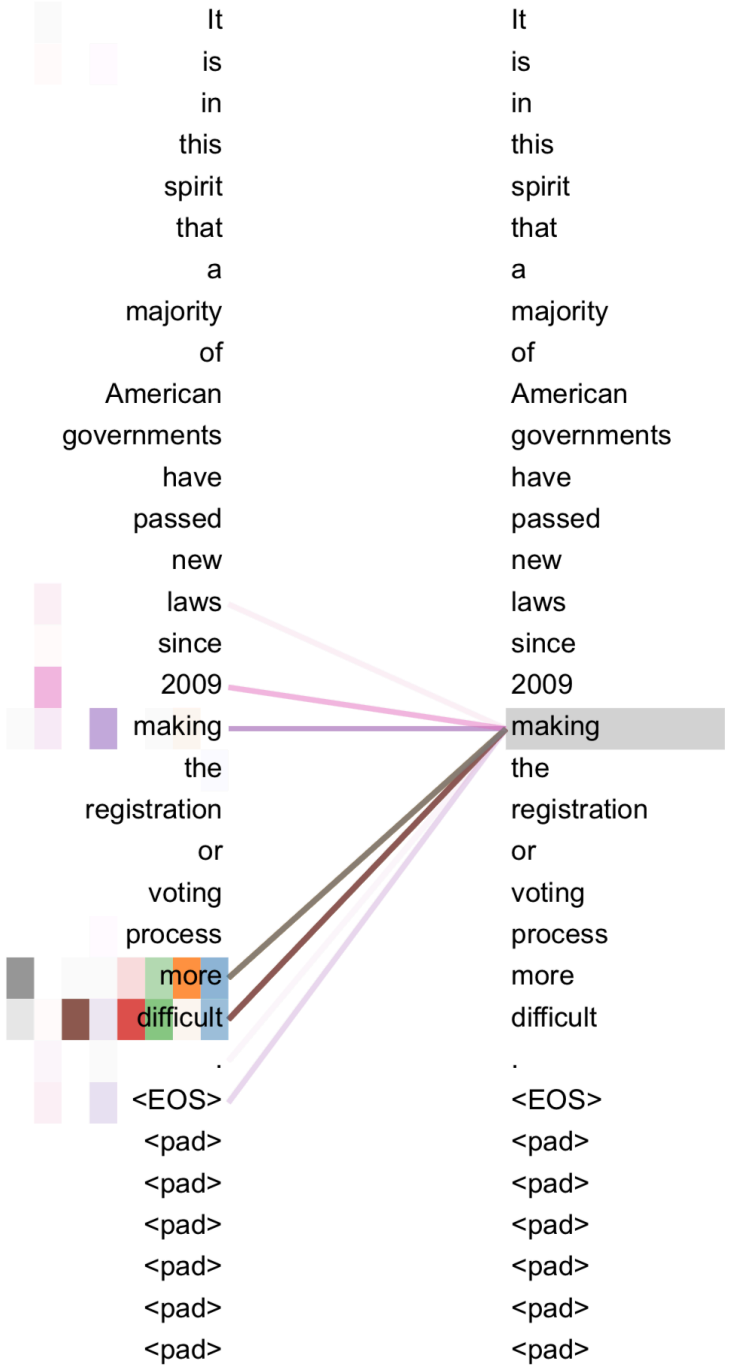
ϵ_{ls} : label smoothing coefficient

Attention Visualization

- Encoder self-attention in layer 5 of 6

- Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’

- Different colors represent different heads.



Attention Visualization

- Two attention heads in layer 5 of 6, involved in coreference resolution

- **Left**: Full attentions for head 5

- **Right**: Isolated attentions for the word ‘its’