Attention & Transformers

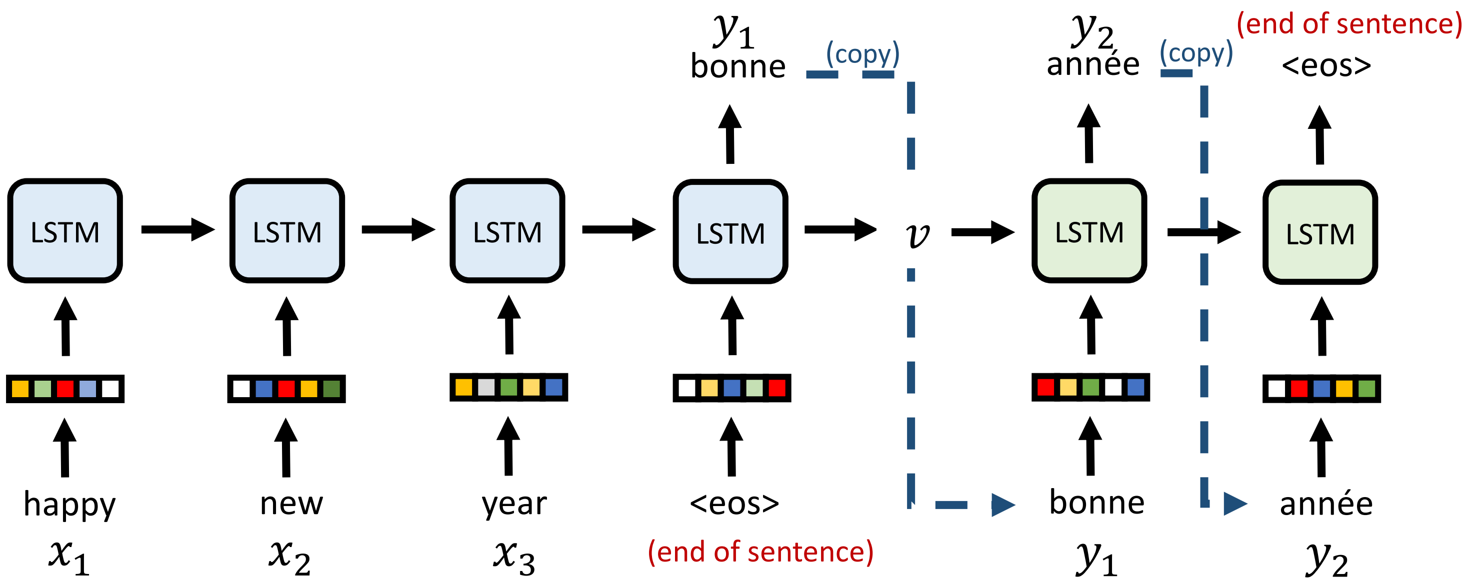
Basics

Yingzhen Li (yingzhen.li@imperial.ac.uk)

Motivation

Recap A Seq2Seq model for machine translation:

What if the sequence is very long?

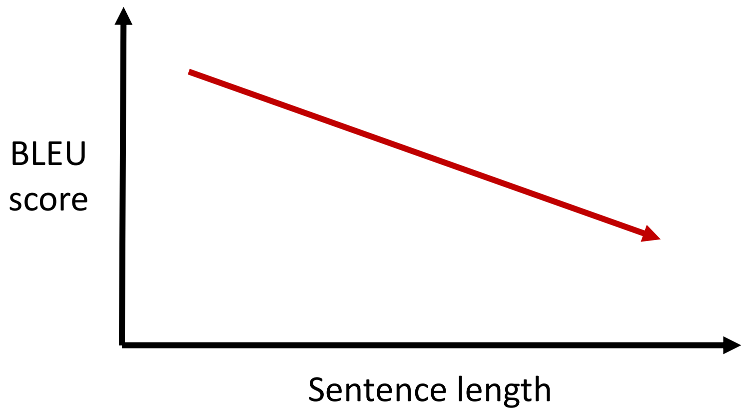


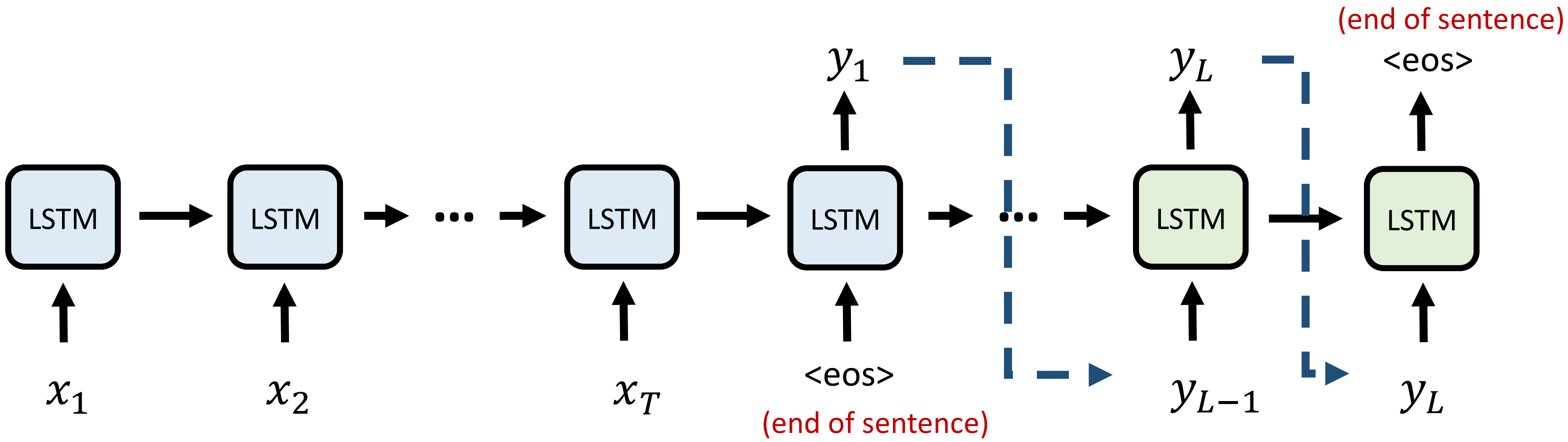
Sutskever et al. Sequence to Sequence Learning with Neural Networks. NeurIPS 2014

Motivation

Recap A Seq2Seq model for machine translation:

What if the sequence is very long?



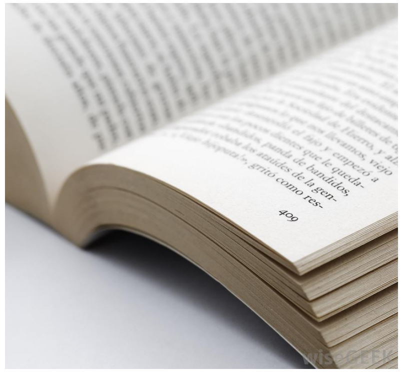


Input:
Since the release of Cyberpunk 2077 on Dec. 10, thousands of gamers have created viral videos featuring a multitude of glitches and bugs — many hilarious — that mar the game. They include tiny trees covering the floors of buildings, tanks falling from the sky and characters standing up, inexplicably pantless, while riding motorcycles...

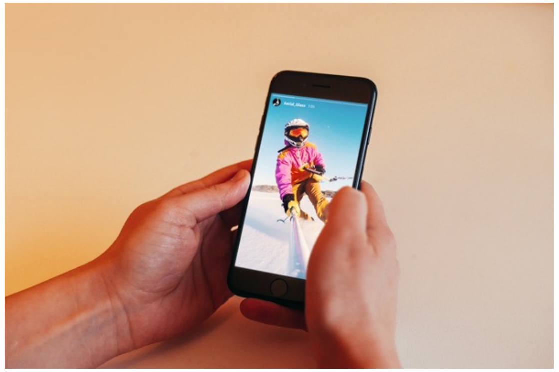
Sutskever et al. Sequence to Sequence Learning with Neural Networks. NeurIPS 2014

Motivation

- Long sequence is everywhere!



A paragraph typically contains hundreds of words



A 30sec short video contains $30 \times 60 = 1{,}800$ frames (60Hz frame rate)

Need efficient ways to handle long-term dependencies!

Attention in *Bahdanau et al.* NMT model

In Seq2Seq model, decoder is defined as

$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, v)$$

Shared representation of the entire input $x_{1:T}$

With attention:

$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, v_l)$$

Each y_l refers to the input sequence differently

$$v_l = \sum_{t=1}^{T} \alpha_{lt} f_t$$

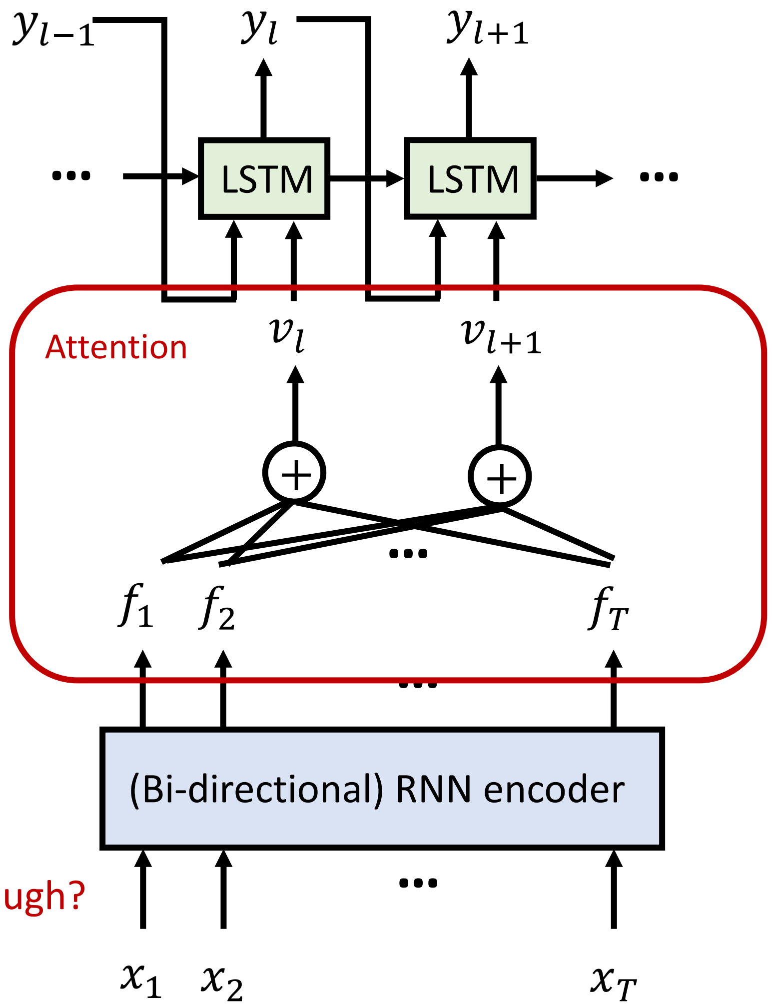
aggregate features by weighted sum

$$\alpha_l = softmax(e_l),\ e_l = (e_{l1}, \dots, e_{lT})$$

$$e_{lt} = a(h_{l-1}^d, f_t)$$

Decoder RNN state at step $l-1$ Encoder feature output at time t

Alignment model $a(\cdot,\cdot)$ score the "similarity/alignment" between two inputs



Still using RNNs for encoder feature extraction -- Can we do attention all the way through?

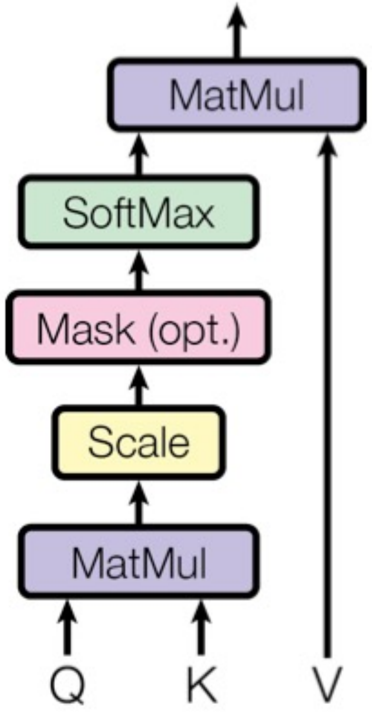
Bahdanau et al. Neural Machine Translation by Jointly Learning to Align and Translate. ICLR 2015

Attention in Transformers

- Single head attention

Attention weights

$$Attention(Q, K, V; a) = \boxed{a\left(\frac{QK^T}{\sqrt{d_q}}\right)} V$$



$Q \in R^{N \times d_q}$: N query inputs, each of dimension d_q
$K \in R^{M \times d_q}$: M key vectors, each of dimension d_q
$V \in R^{M \times d_v}$: M value vectors, each of dimension d_v
$a(\cdot)$: activation function applied row-wise

Self attention: $K = Q$

Vaswani et al. Attention is all you need. NeurIPS 2017

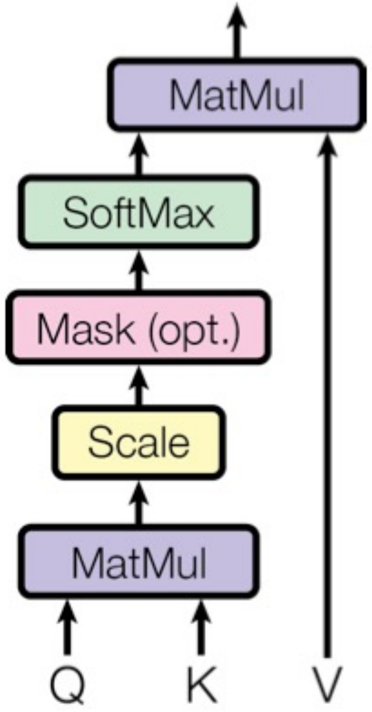
Attention in Transformers

- Single head attention

$(QK^T)_{ij} = \langle q_i, k_j \rangle$: similarity between query q_i and key k_j

$$Attention(Q, K, V; a) = \underline{a\left(\frac{QK^T}{\sqrt{d_q}}\right)}V$$

Retrieve values according to the weighting



Hard attention: $a(x) = onehot(argmax\ x_i)$ for $x = (x_1, \dots, x_d)$

- Each key vector k_i is associated with a value vector v_i $(K = (k_1, \dots, k_M)^T, V = (v_1, \dots, v_M)^T)$
- The query matrix packs the query vectors $Q = (q_1, \dots, q_N)^T$
- $a(\cdot)$ applied row-wise: the n^{th} row of the output is
  $onehot(argmax\ \langle q_n, k_i \rangle)$
- Therefore, the n^{th} row of the attention output will be v_{i_n} for $i_n = argmax\ \langle q_n, k_i \rangle$

Vaswani et al. Attention is all you need. NeurIPS 2017

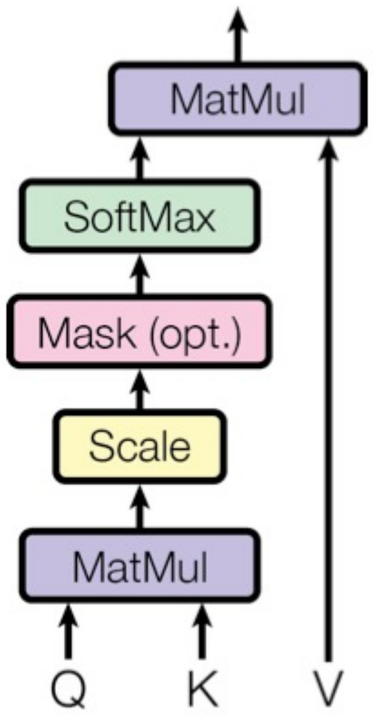
Attention in Transformers

- Single head attention

$(QK^T)_{ij} = \langle q_i, k_j \rangle$: similarity between query q_i and key k_j

$$Attention(Q, K, V; a) = \underline{a\left(\frac{QK^T}{\sqrt{d_q}}\right)} V$$

Retrieve values according to the weighting



Soft attention: $a(x) = softmax(x)$ for $x = (x_1, \ldots, x_d)$

- Each key vector k_i is associated with a value vector v_i $(K = (k_1, \ldots, k_M)^T, V = (v_1, \ldots, v_M)^T)$
- The query matrix packs the query vectors $Q = (q_1, \ldots, q_N)^T$
- $a(\cdot)$ applied row-wise: the n^{th} row of the output is

  $softmax(q_n K^T)$
- Therefore, the n^{th} row of the attention output will be a weighted sum of value vectors v_j with weighting proportional to $\exp(\langle q_n, k_j \rangle)$

Vaswani et al. Attention is all you need. NeurIPS 2017

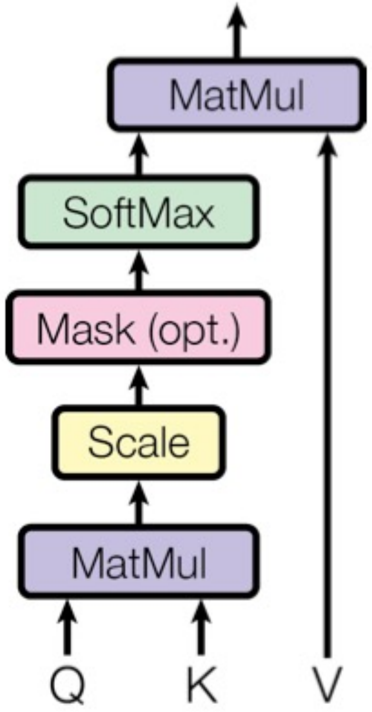
Attention in Transformers

- Single head attention

$(QK^T)_{ij} = \langle q_i, k_j \rangle$: similarity between query q_i and key k_j

$$MaskedAttention(Q, K, V; a, M) = a\left(mask\left(\frac{QK^T}{\sqrt{d_q}}, M\right)\right)V$$

Retrieve values according to the weighting and masking



Masked attention: mask out some of the attention values,
Example when $a(\cdot)$ is softmax:

- M_{nm} takes values 0 (mask out) or 1 (keep in)
- With $M_{nm} = 0$, set $(QK^T)_{nm} = -\infty$ so that $\exp\left(\frac{QK^T}{\sqrt{d}}\right)_{nm} = 0$
- So value v_m will NOT contribute to the attention output for query q_n
- Useful for sequence prediction with a given ordering: in test time, "future" is not available for the "current" to attend

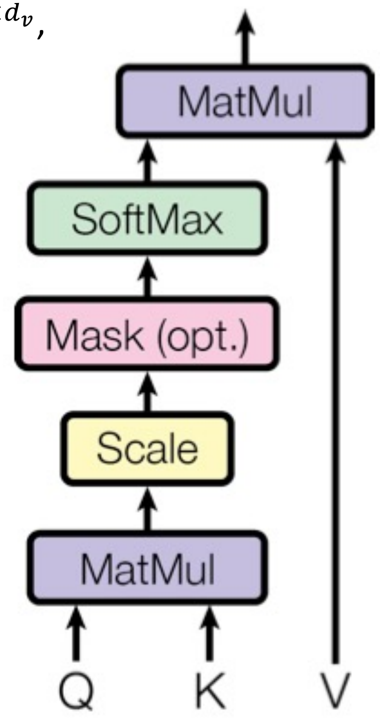
Vaswani et al. Attention is all you need. NeurIPS 2017

Attention in Transformers

- Single head attention

$Q \in R^{N \times d_q}, K \in R^{M \times d_q}, V \in R^{M \times d_v}$,
$a(\cdot)$ applied row-wise

$$Attention(Q, K, V; a) = a\left(\frac{QK^T}{\sqrt{d_q}}\right)V$$



Complexity analysis:

- Time complexity: $O(MNd_q + MNd_v)$
- Space complexity: $O(MN + Nd_v)$ (incl. intermediate steps)
- Parameters to learn:
  - K, V in the usual form: $O(Md_q + Md_v)$
  - V only for self attention: $O(Nd_v)$
  - Can also use $V = K$ (meaning $Q = V = K$ in self attention)

Vaswani et al. Attention is all you need. NeurIPS 2017

Attention in Transformers

- Multi-head attention

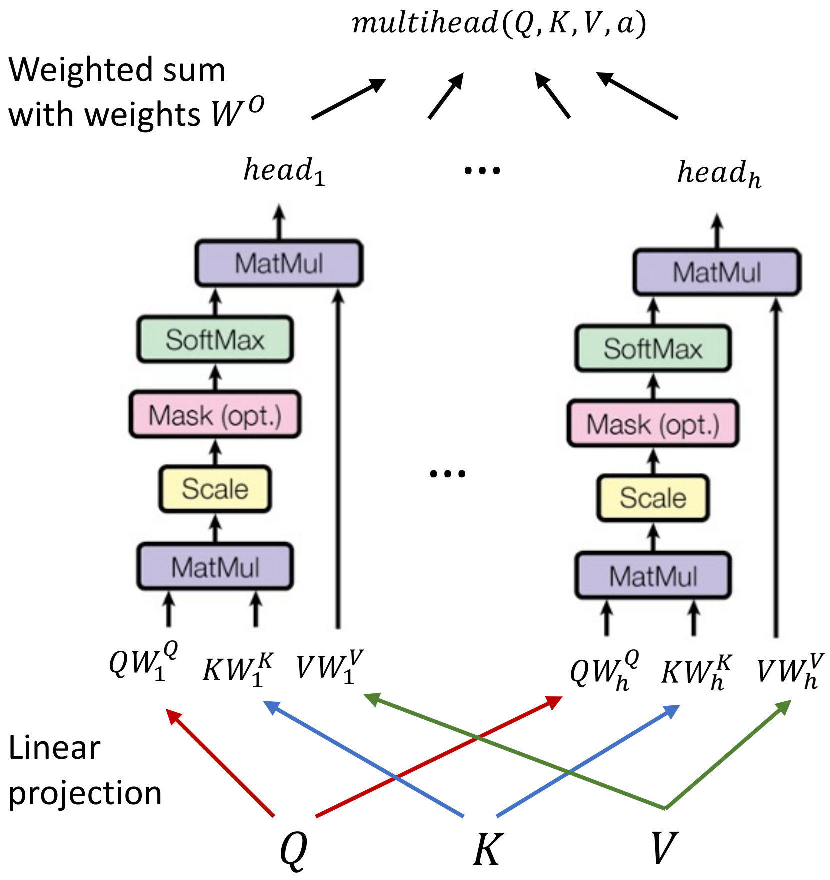
$$Multihead(Q, K, V, a) = concat(head_1, \dots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V, a)$$

$$Attention(Q, K, V; a) = a\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Different head represent different alignments, e.g.
when each query represents a word and self-attention is used:

- Head 1: find keys that are semantically similar to q
- Head 2: find keys that makes (q, k) as a subject-verb pair
- …



Vaswani et al. Attention is all you need. NeurIPS 2017

Attention in Transformers

- Multi-head attention

$$Multihead(Q, K, V, a) = concat(head_1, \ldots, head_h)W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V, a)$$

Complexity analysis (assume $Q \in R^{N \times d_q}$ projected to $R^{N \times \tilde{d}_q}$ and so on):

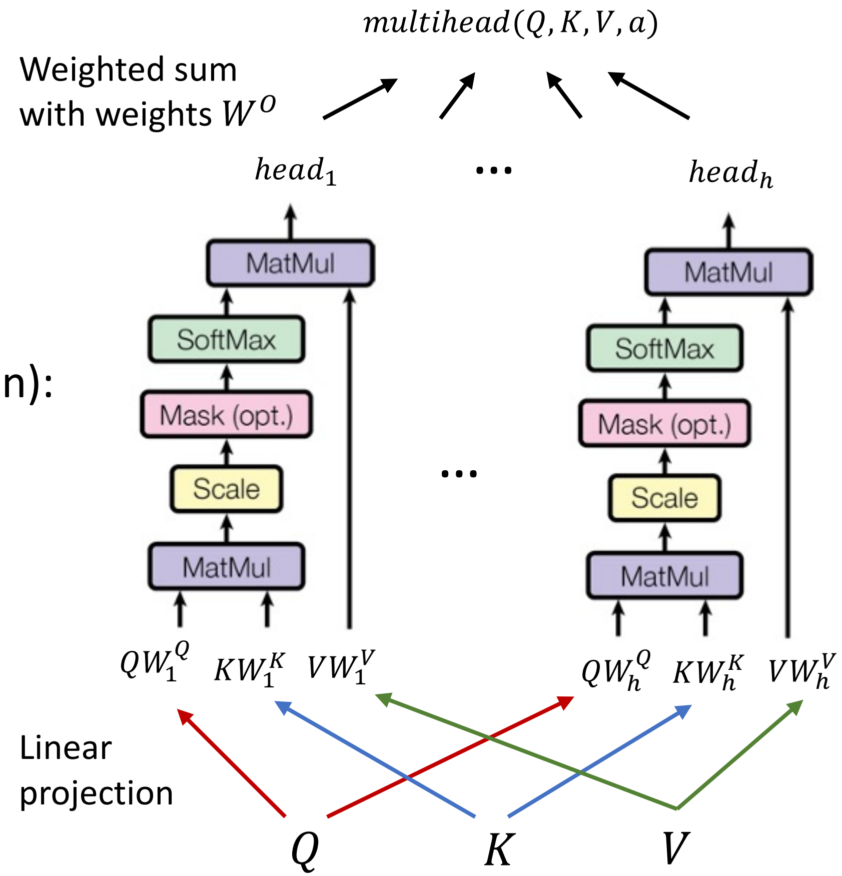
- Time complexity:

$$O(\underbrace{hMN(\tilde{d}_q + \tilde{d}_v)}^{\text{Attention heads}} + \overbrace{h(\tilde{d}_q d_q(M+N) + \tilde{d}_v d_v M)}^{\text{projections}} + \overbrace{Nh\tilde{d}_v d_{out}}^{\text{combined output}})$$

- Space complexity:

$$O(\overbrace{hN(M + \tilde{d}_v)}^{\text{Attention heads}} + \overbrace{h\left((N+M)\tilde{d}_q + M\tilde{d}_v\right)}^{\text{projections}} + \overbrace{Nd_{out}}^{\text{combined output}})$$

- Parameters to learn (apart from K and V):
  - Projection parameters W_i^Q, W_i^K, W_i^V
  - Output weight matrix W^O



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

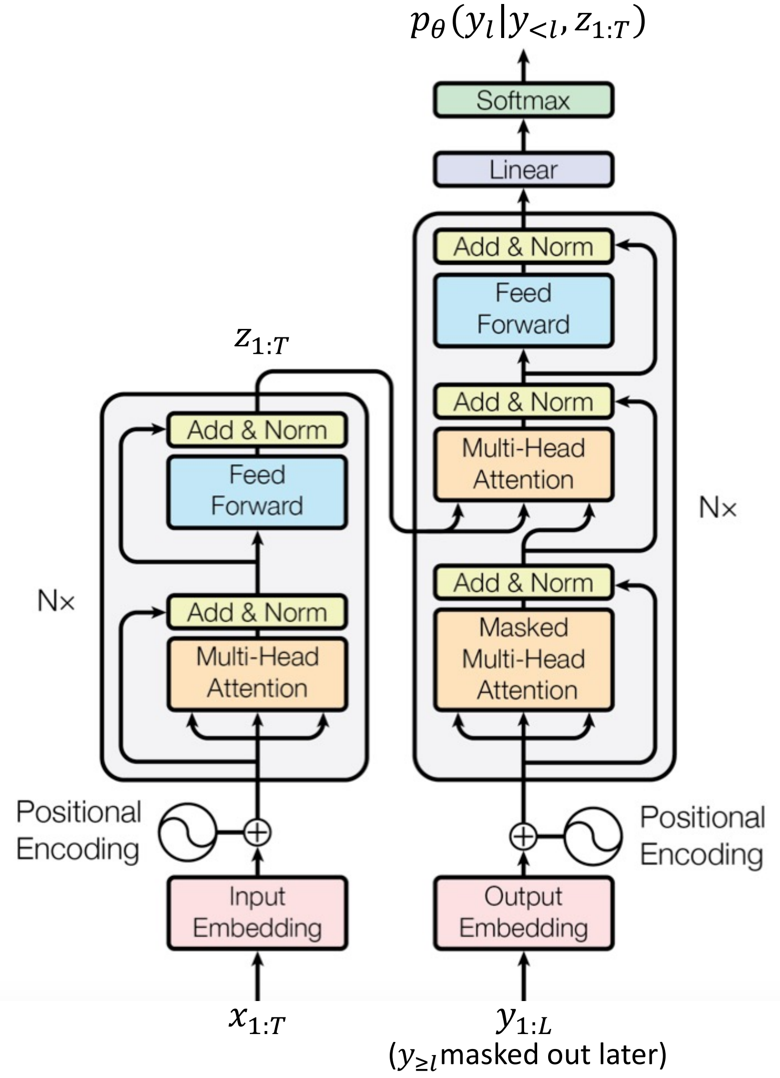
$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

Encoder attention outputs used in decoder

The input to the decoder:

- Training time: $y_{1:L}$
- Test time: $(y_1, \dots, y_{l-1}, \emptyset, \dots, \emptyset)$



Vaswani et al. Attention is all you need. NeurIPS 2017

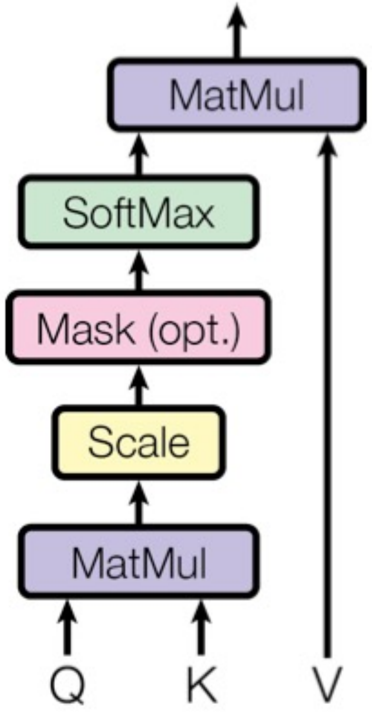
Transformer Architecture

- Single head attention

$$Attention(Q, K, V; a) = a\left(\frac{QK^T}{\sqrt{d_q}}\right)V$$

Permutation equivariant:

- $\tilde{Q}$ constructed by swapping the i^{th} and j^{th} row in Q
  $\Rightarrow Attention(\tilde{Q}, K, V, a)$ equals to $Attention(Q, K, V, a)$
  except that the i^{th} and j^{th} rows are swapped
- The ordering information is irrelevant!



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

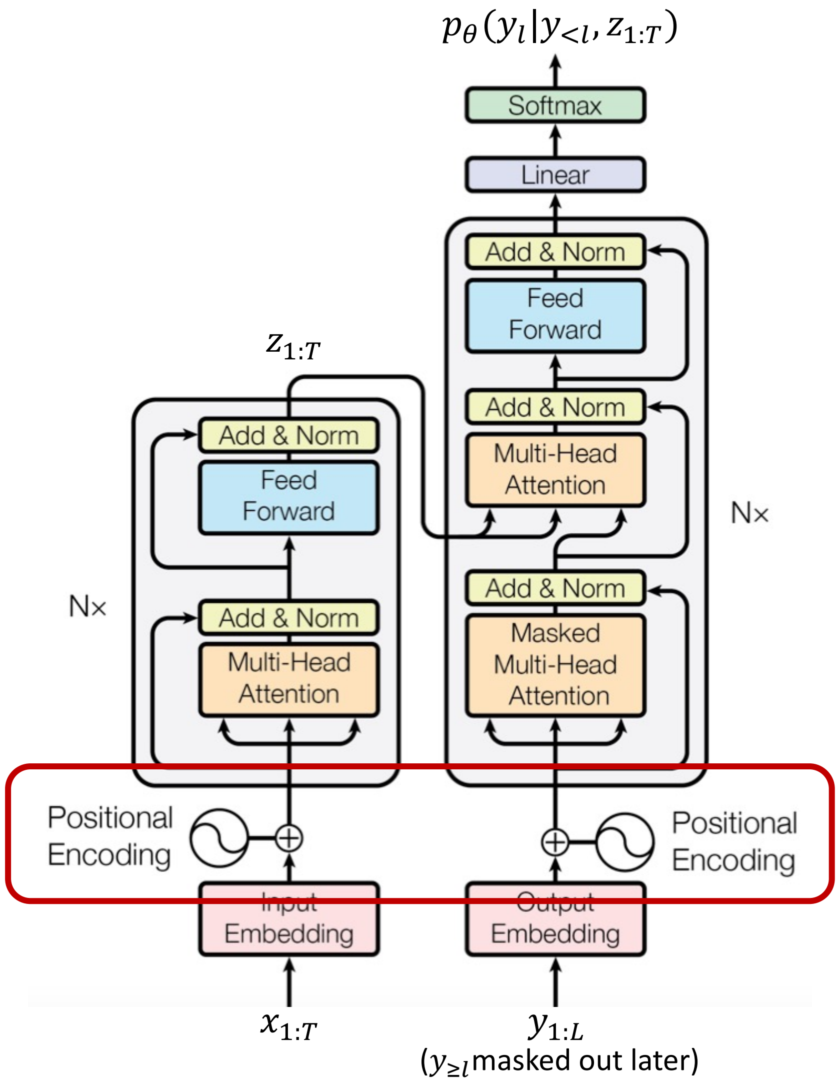
Encoder attention outputs used in decoder

Position encoding: inject ordering information
Can either be learned or be a pre-defined mapping, e.g.:

$$PE(pos, 2i) = \sin(pos/10000^{2i/d_{out}})$$

$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{out}})$$

output: $word\ embedding(x_t) + PE(t, 1{:}d_{emb})$



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

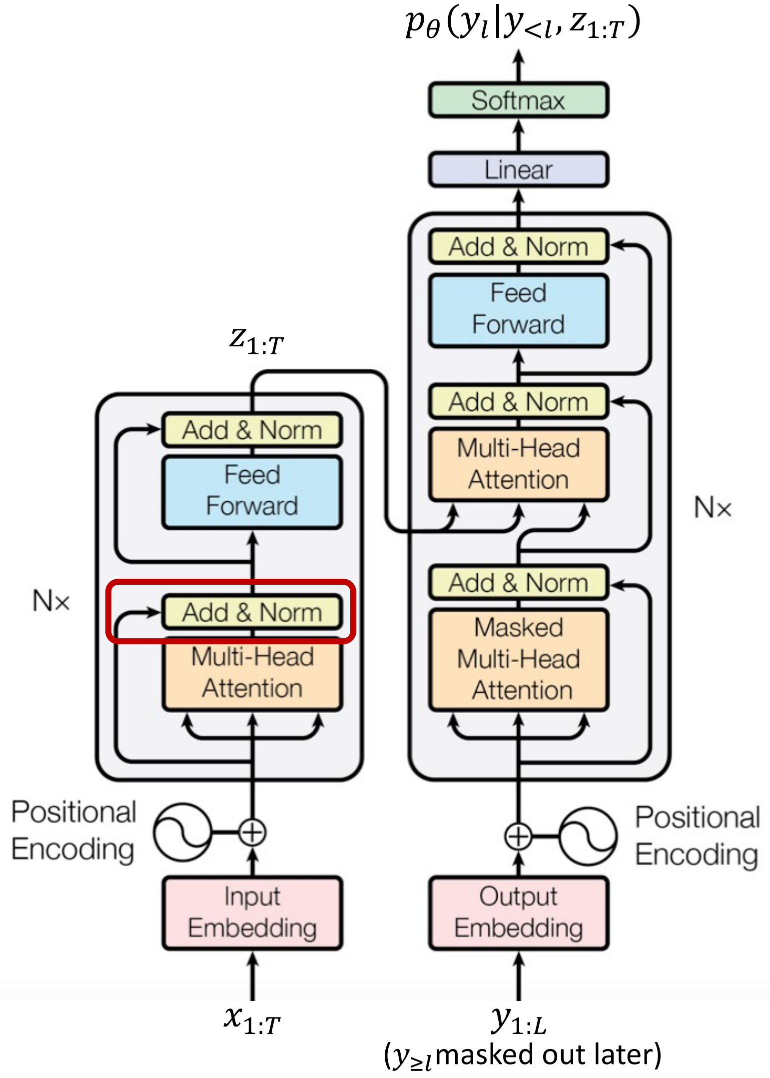
Encoder attention outputs used in decoder

Add & Norm:

Residual connection

$$LayerNorm(x + Sublayer(x))$$

Layer normalization is similar to batch normalization except that it is performed within a single hidden layer output



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

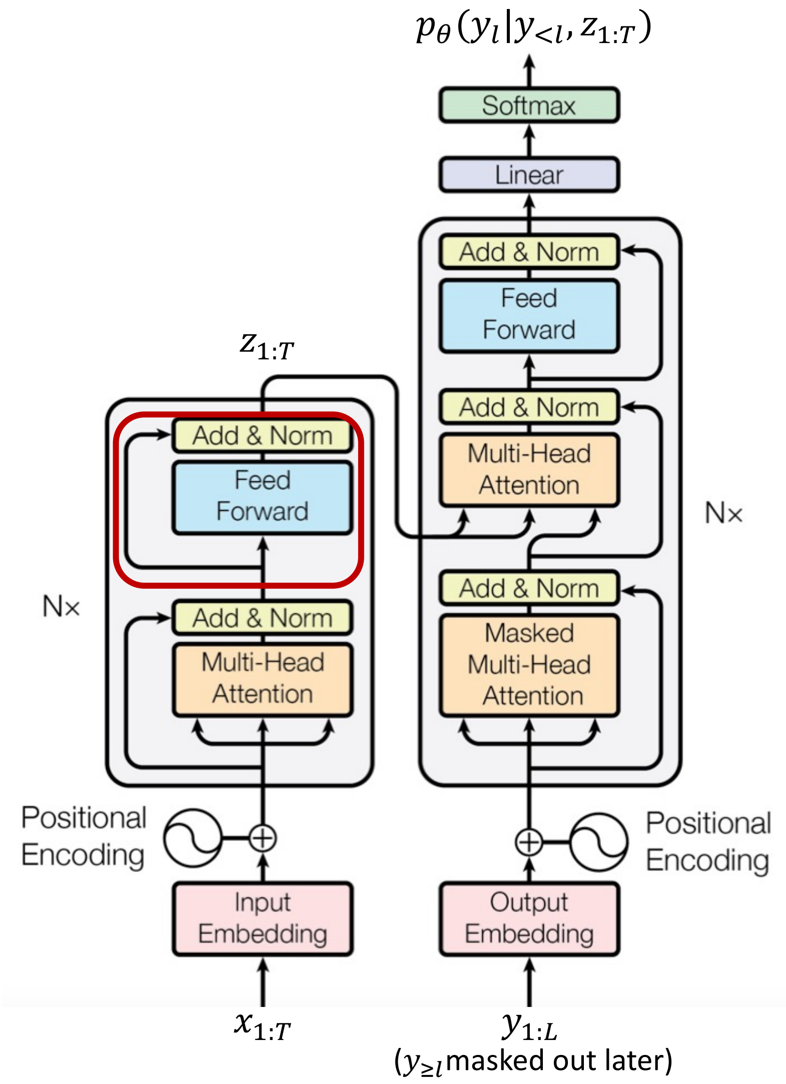
$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

Encoder attention outputs used in decoder

Feed-forward network:

Applied to each of the output value vectors (i.e. row vectors) independently and identically



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

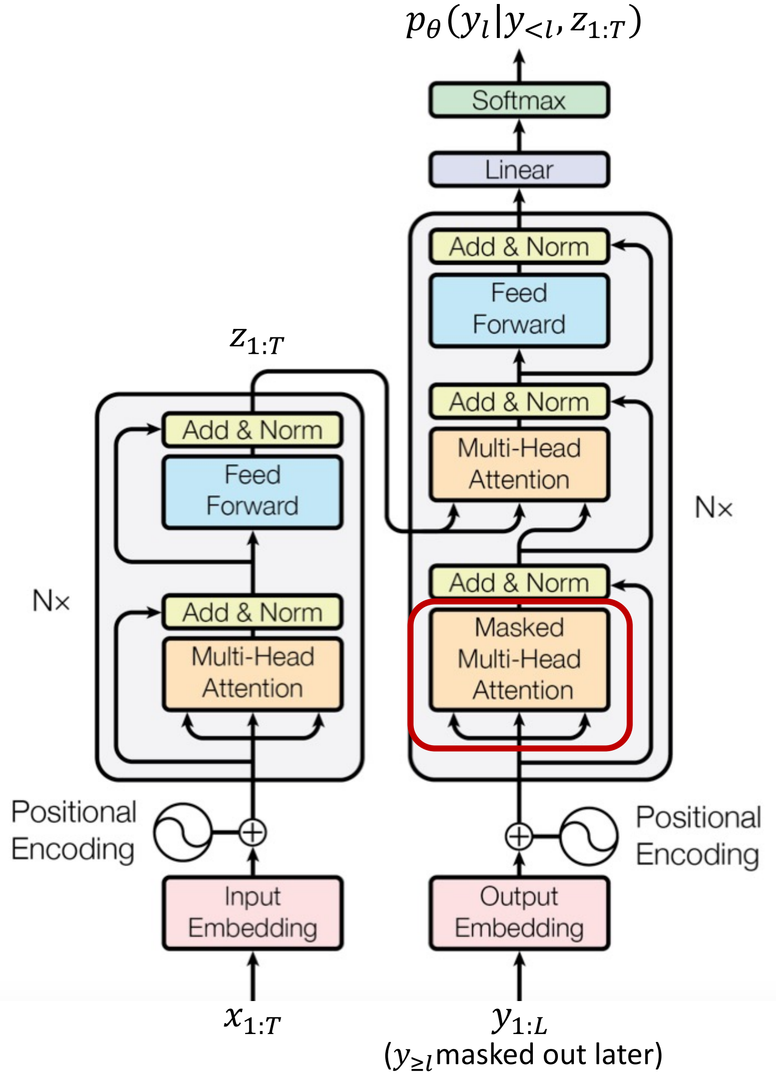
$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

Encoder attention outputs used in decoder

Maked Multi-head Attention:
Prevent the model to use "future" information for predicting the current output
(training time input: $(y_1, \dots, y_{l-1}, , y_l \dots, y_L)$)
(test time input: $(y_1, \dots, y_{l-1}, \emptyset, \dots, \emptyset)$)

should be masked out



Vaswani et al. Attention is all you need. NeurIPS 2017

Transformer Architecture

Attention based encoder + decoder:

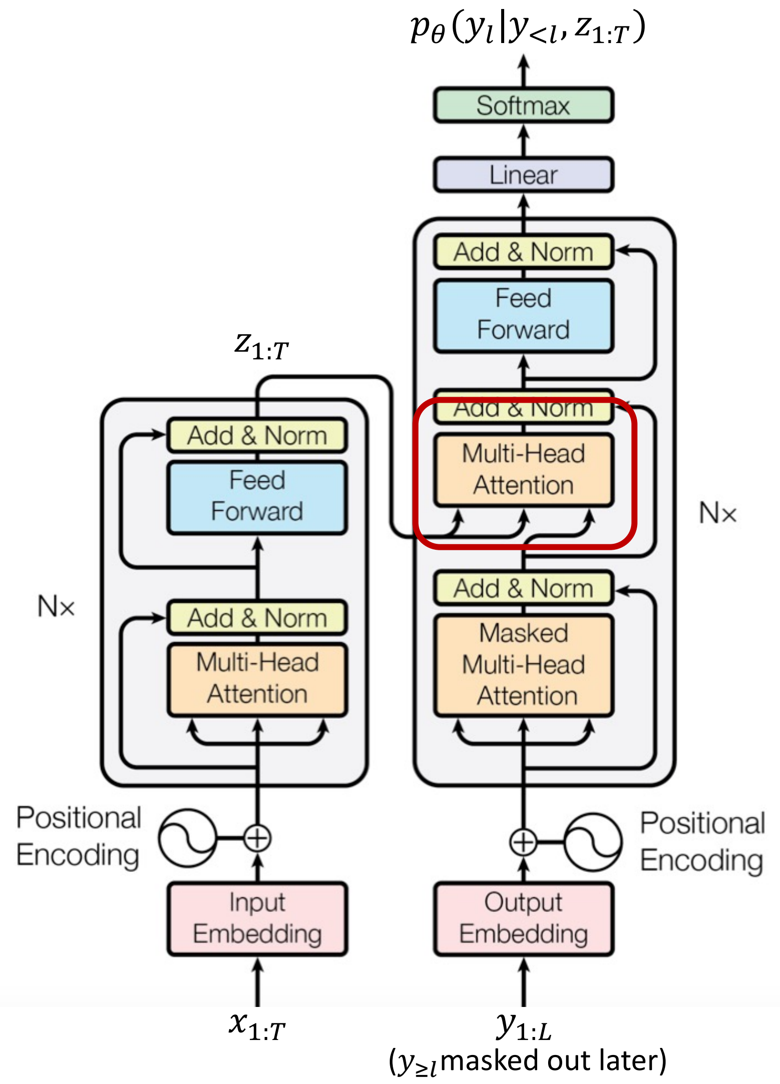
$$p_\theta(y_{1:L}|x_{1:T}) = \prod_{l=1}^{L} p_\theta(y_l|y_{<l}, z_{1:T})$$

$y_{\geq l}$ will be masked out in the first layer of decoder

Encoder attention outputs used in decoder

Multi-head Attention using encoder output $z_{1:T}$

- $z_{1:T}$ are used as the keys and values of this attention module
- Allow the decoder to attend every word in the input $x_{1:T}$ for each of the predicted output $y_{1:l-1}$ so far

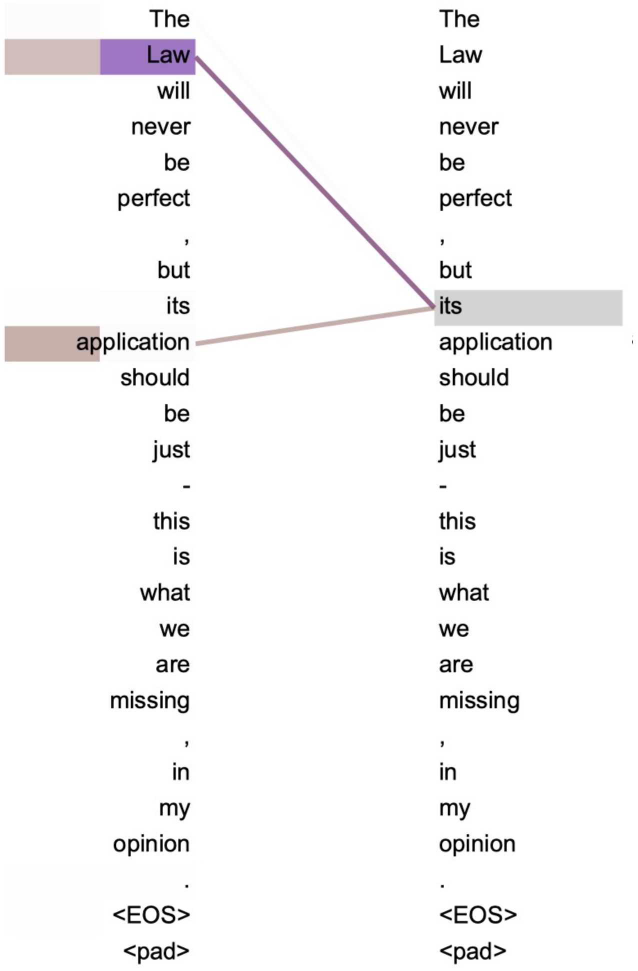


Vaswani et al. Attention is all you need. NeurIPS 2017

Visualising Learned Attentions

reflect structure
of the sentence

Different head
show different
patterns



Vaswani et al. Attention is all you need. NeurIPS 2017