Meeting Note

Author: Henry Lin
Date: 2024/09/09 – 2024/09/13

Weekly Summary
Attention Is All You Need [1].

Plan for Next Week
High-Resolution Image Synthesis with Latent Diffusion Models (arXiv: 2112.10752)

Details
Traditional approaches such as recurrent neural networks (RNNs) and long short-term memory (LSTM) [2] models have been successful in the realm of natural language processing (NLP), but they have a serious drawback on computational efficiency since the outputs have to be computed recurrently. Other approaches, such as ByteNet [3] and ConvS2S [4], utilize convolutional neural networks (CNNs) [5] as the backbone of the models to address this problem. However, these architectures face another significant challenge: they struggle to capture the dependencies between distant tokens effectively. The authors proposed a model based on the attention mechanism called Transformer, which avoids the recurrence and allows more parallelization, furthermore, the attention mechanism enables the model to consider the relations between tokens across the entire sequence.

Most competitive models follow the encoder-decoder architecture, where the encoder maps a sequence of input x to a latent space representation z, and the decoder generates the output y based on given z. The authors design the Transformer based on this architecture, as shown in Figure 1. Both the encoder and the decoder are composed of a stack of $N = 6$, and all embedding layers and sub-layers have the dimension $d_{model} = 512$, the feed-forward modules are fully-connected layers with residual connections [6]. In addition, the second multi-head attention module in the decoder, which integrates the encoder with the decoder, takes the outputs of encoder as the key and value, and the outputs of the first multi-head attention module as the query.

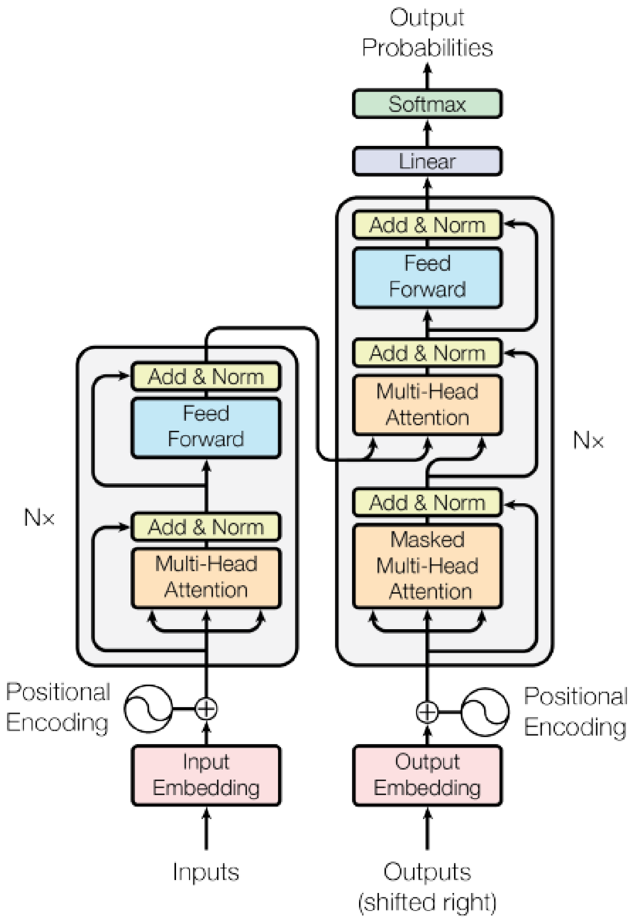

Figure 1. The architecture of Transformer model.

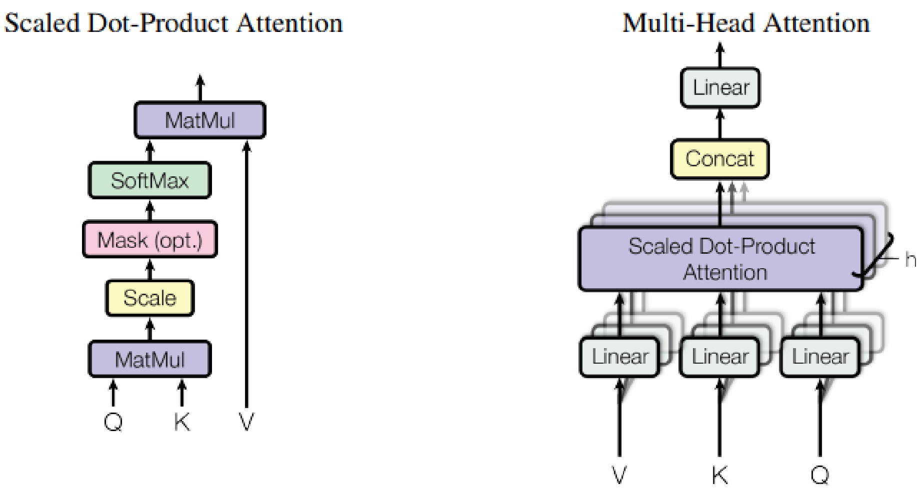

Figure 2. (left) Scaled Dot-Product Attention [Re 1]. (right) Multi-Head Attention.

There are two commonly used attention functions, that is, additive (Bahdanau-style) attention [9, Re 2] and dot-product (multi-plicative, Luong-style) attention [10], both has the similar theoretical complexity. Since the dot-product attention is faster and more space-efficient in practice, the authors determine to use it. Here is the mathematical representation of scaled dot-product attention (see code scaled_dot_product_attention.py):

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

where Q, K, V denote query, key and value vectors with dimensions d_k, d_k, d_v respectively [Re 3]. Notably, the experiments [Re 4] show that the additive attention outperforms dot-product attention for the small values of d_k.

Further experiments [Re 4] show that instead of using a single attention function, using learnable linear projectors to map Q, K, V to vectors with dimensions d_k, d_k, d_v respectively (same with the original dimensions), as shown in Figure 2, lead to a better result. The attention functions then parallelly performed on the outputted vectors as follow (see code multihead_attention.py):

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

where the $W_i^Q \in \mathcal{R}^{d_{model} \times d_k}$, $W_i^K \in \mathcal{R}^{d_{model} \times d_k}$, $W_i^V \in \mathcal{R}^{d_{model} \times d_v}$, and $W^O \in \mathcal{R}^{hd_v \times d_{model}}$ are metrics with learnable parameters, and h=8 denotes the attention functions in parallel. Multi-head attention allows the model to learn information from different subspaces at different positions jointly.

Unlike CNNs or RNN, Transformer is unable to capture the order of the sequence since the embeddings are inputted parallelly. To address this problem, the authors add the position encodings [Re 5] to the input embeddings with the following functions to ensure that the position information is considered during processing:

$$\begin{aligned}PE_{(pos,2i)} &= sin(pos/10000^{2i/d_{\mathrm{model}}}) \\ PE_{(pos,2i+1)} &= cos(pos/10000^{2i/d_{\mathrm{model}}})\end{aligned}$$

where pos is the position and i is the dimension. After the position encoding PE is calculated, it is added to the word embeddings (each token's embedding is summed with its corresponding positional encoding) and then input into the model.

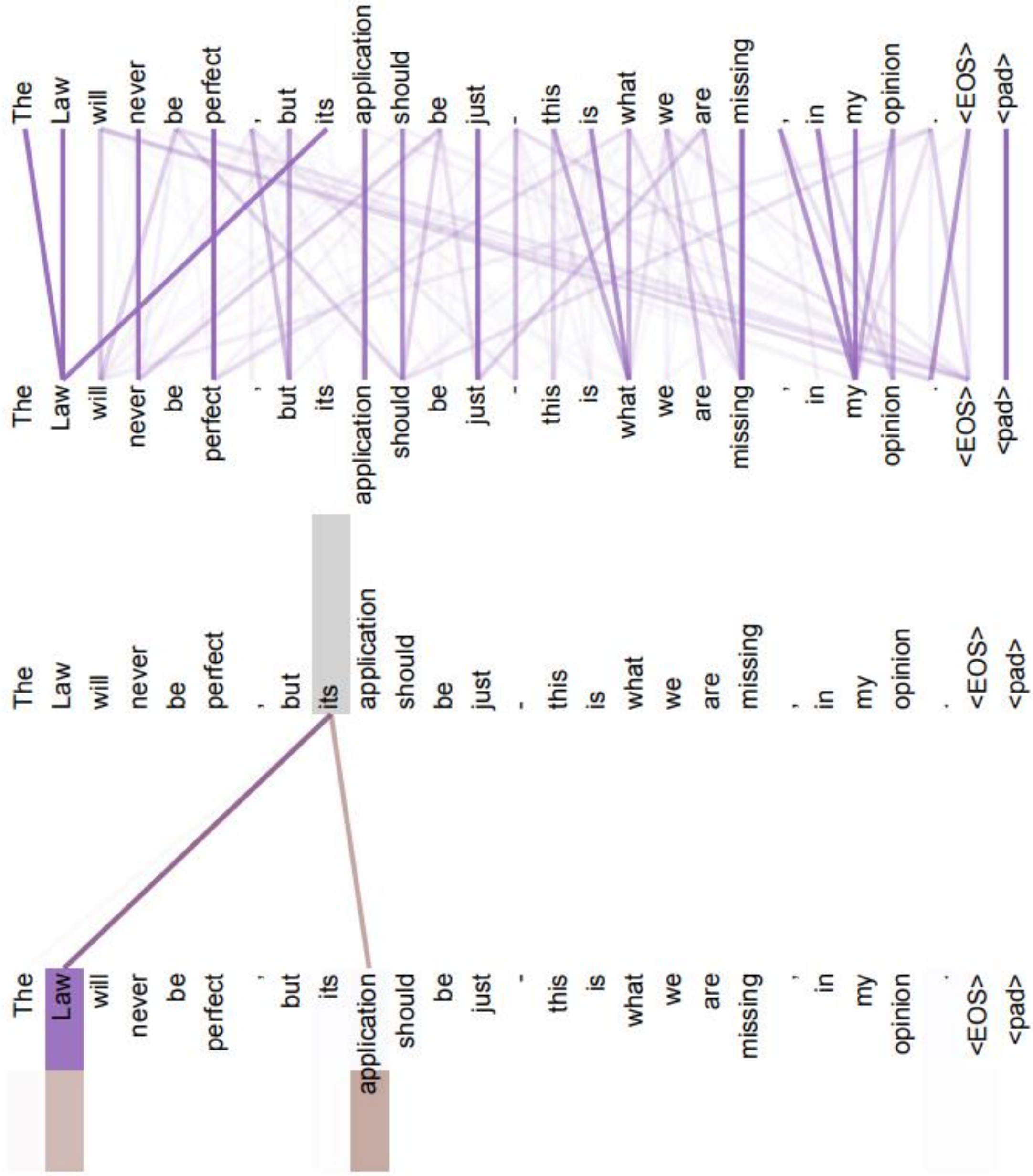

Figure 3. The visualized attention distributions.

And there are three reasons why the authors choose self-attention instead of convolution and recurrence mechanism: (1) the computational complexity is lower; (2) the capability of parallelization (since the attention function utilities the highly optimized matrix multiplication code) is higher; (3) path length between distant dependencies in network is constant. In addition, self-attention also provides more interpretability, as the attention can be visualized as Figure 3.

Table 1. The comparisons of complexities, sequential operations, and maximum path length in different layer types.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Table 2. Experimental results with different hyper-parameters.

	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

Table 3. Training cost and experimental results on BLEU [12] benchmark.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet	23.75			
Deep-Att + PosUnk		39.2		$1.0 \cdot 10^{20}$
GNMT + RL	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Remarks

[Re 1]

"MatMul" denotes matrix multiplication, which can be done with @ in PyTorch.

[Re 2]

Another paper titled "Fastformer: Additive Attention Can Be All You Need" [8] released on 2021 utilities the additive attention instead of dot-product attention to reduce the computational complexity.

[Re 3]

The only different between dot-product attention and the original multi-plicative attention is the scaling factor $\frac{1}{\sqrt{d_k}}$. The authors scale the dot product by $\frac{1}{\sqrt{d_k}}$ since that they suspect for large d_k, the dot products grow very large because the dot product is a summation of d_k which can be written as $q \cdot k = \sum_{i=1}^{d_k} q_i k_i$. When it becomes very large and passed thought the softmax function, the output of softmax function will be close to either 0 or 1, which leads the gradients become extremely small. That is the reason why it called "Scaled Dot-Product Attention".

The other question is that why they use $\frac{1}{\sqrt{d_k}}$ instead of $\frac{1}{d_k}$? The selection of $\frac{1}{\sqrt{d_k}}$ ensures that the attention mechanism can work well in a variety of dimensions without interfering with gradient flow during training by finding a compromise between numerical stability and standardizing the dot product's magnitude. $\frac{1}{d_k}$ would likely lead to excessively small values and poor gradient behavior.

[Re 4]

The results of these experiments are not provided.

[Re 5]

The concept of "position encoding (position embedding)" seems to be first proposed by Gehring et al. [10]. They use absolute position to equip the model with a sense of order.

Besides the proposed position encoding, there are many other position encoding functions, some of which even include learnable parameters. The experimental result of learnable position encoding is shown in Table 2 row (E).

[Re 6]
There are also many variations of Transformer which can be found at https://wikidocs.net/167210.

References

[1] VASWANI, Ashish. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
[2] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long short-term memory. *Neural computation*, 1997, 9.8: 1735-1780.
[3] KALCHBRENNER, Nal, et al. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.
[4] GEHRING, Jonas, et al. Convolutional sequence to sequence learning. In: *International conference on machine learning*. PMLR, 2017. p. 1243-1252.
[5] O'SHEA, Keiron; NASH, Ryan. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.
[6] HE, Kaiming, et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 770-778.
[7] BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
[8] WU, Chuhan, et al. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*, 2021.
[9] BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
[10] LUONG, Minh-Thang; PHAM, Hieu; MANNING, Christopher D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.
[11] GEHRING, Jonas, et al. Convolutional sequence to sequence learning. In: *International conference on machine learning*. PMLR, 2017. p. 1243-1252.
[12] PAPINENI, Kishore, et al. Bleu: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting of the Association*

for Computational Linguistics. 2002. p. 311-318.

Comments

This week, we improved our meeting note by:

(1) deciding to use “ (U+201C) and ” (U+201D) instead of " (U+0022) as the quotation marks, as they are considered more professional and polished. To maintain consistency, the use of U+0022 will be discontinued.

(2) determining the docstring style for code implementations should be Numpy-Style; the types of function arguments and return values should be clearly indicated; the name of variable should convey its purpose clearly, and the type can be indicated optionally; the docstring should contains four sections as follow: descriptions, parameters, returns, and notes; required modules should be imported clearly.