(2017NIPS)Attention Is All You Need

Paper Link

Abstract

The dominant sequence transduction models are based on complex **recurrent** or **convolutional neural networks** that include an **encoder** and a **decoder**.

The best performing models also connect the encoder and decoder through an **attention mechanism**.

主流的序列转换模型基于复杂的**递归**或**卷积神经网络**，其中包括一个**编码器**和一个**解码器**。性能最好的模型还通过**注意力机制**将编码器和解码器连接起来。

We propose a new simple network architecture, the **Transformer**, based solely on attention mechanisms, **dispensing with recurrence and convolutions entirely**.

我们提出了一种新的简单网络架构--**Transformer**，它完全基于注意力机制，**无需递归和卷积**。

1. Introduction

Recurrent model aligns the positions to steps in computation time, they generate a sequence of **hidden states** h_t, as a function of the **previous hidden state** h_{t-1} and the **input for position t**.

递归模型将位置与计算时间的步长对齐，它们会生成隐藏状态 h_t的序列，作为前一个隐藏状态 h_{t-1} 和位置 t 的输入的函数。

This inherently sequential nature precludes **parallelization within training examples**, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples.

这种固有的序列性质排除了**训练实例内的并行化**，而在序列长度较长时，这一点变得至关重要，因为内存约束限制了跨实例的批处理。

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences.

注意力机制已成为引人注目的序列建模和转导模型的一个组成部分，它可以对依赖关系进行建模，而无需考虑其在输入或输出序列中的距离。

In this work we propose the **Transformer**, a model architecture eschewing recurrence and instead **relying entirely on an attention mechanism** to draw global dependencies between input and output.

在这项工作中，我们提出了 **Transformer** 模型架构，它摒弃了递归，而是**完全依赖注意力机制**来绘制输入和输出之间的全局依赖关系。

2. Background

The number of operations required to relate signals from two arbitrary input or output positions grows in the distance between positions.

This makes it more difficult to learn **dependencies between distant positions**.

将两个任意输入或输出位置的信号联系起来所需的运算次数随位置间距离的增加而增加。

这就增加了学习**远距离位置间依赖关系**的难度。

In the **Transformer** this is **reduced to a constant number of operations**, albeit at the cost of reduced effective resolution due to averaging attention-weighted positions, an effect we counteract with **Multi-Head Attention**.

在 **Transformer** 中，这将被减少到一个**恒定的操作数**，但代价是由于平均注意力加权位置而降低了有效分辨率，但**多头注意力**能够抵消这一影响。

Self-attention, sometimes called **intra-attention** is an attention mechanism relating different positions of a single sequence in order to compute a representation of the sequence.

自我注意（有时也称为**内部注意**）是一种注意机制，它将单个序列的不同位置联系起来，以计算序列的表征。

The **Transformer** is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution.

Transformer 是第一个完全依靠自我注意来计算输入和输出表示而不使用序列对齐 RNN 或卷积的转换模型。

3. Model Architecture

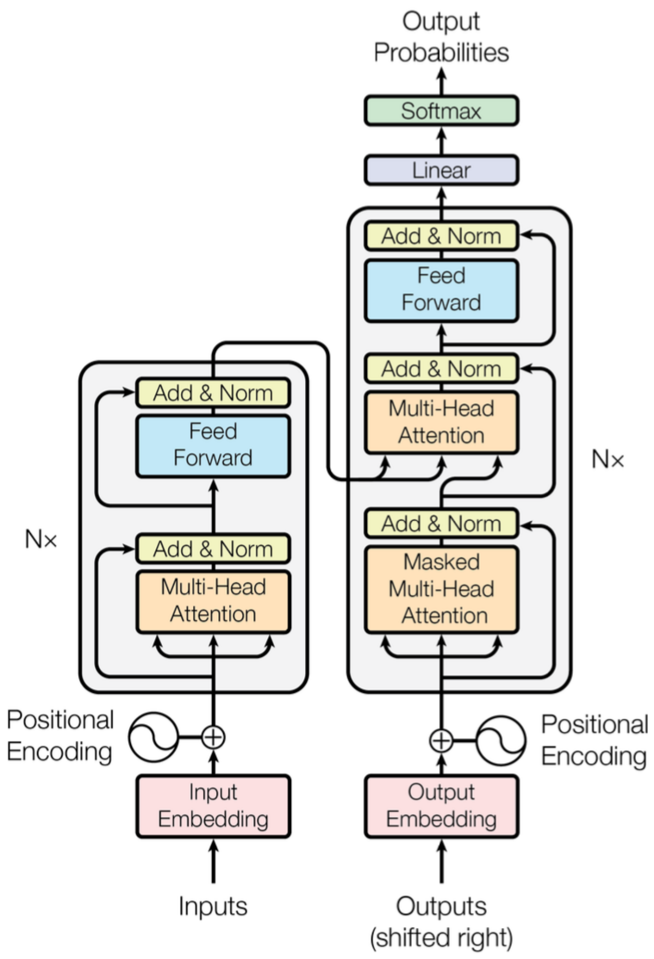
Encoder : $(x_1, \ldots x_n) \rightarrow (z_1, \ldots, z_n)$

Middle $(z_1, \ldots, z_n) \rightarrow (z_1, \ldots, z_n)$

Decoder: $(z_1, \ldots, z_n) \rightarrow (y_1, \ldots, y_n)$

Recurrent model at each step the model is **auto-regressive**, consuming the previously generated symbols as additional input when generating the next.

递归模型在每一步中，模型都是**自动回归**的，在生成下一步时，会消耗之前生成的符号作为额外输入。



Encoder(left) and **Decoder**(right) using stacked self-attention and point-wise, fully connected layers.

编码器（左侧）与**解码器**（右侧）使用了堆叠式自关注和点式全连接层的整体架构。

3.1. Encoder and Decoder Stacks

Encoder:

- 6 identical layers, each layer has 2 sub-layers
- first sub-layer is multi-head self-attention mechanism, the second is a simple, positionwise fully connected feed-forward network.
- employ a residual connection around each of the two sub-layers, followed by layer normalization. The output of each sub-layer is $LayerNorm(x + SubLayer(x))$
- the output dimension of all layers is 512

编码器:

- 6个相同的层，每层都有2个子层
- 第一个是多头自注意机制，第二个是简单的位置全连接前馈网络。
- 两个子层的每个子层周围采用残差连接，并进行归一化。每个子层的输出为 $LayerNorm(x + SubLayer(x))$
- 每个层的输出维数为512

Decoder:

- 6 identical layers, 2 sub-layers as same as **Encoder**, a third sub-layer performs multi-head attention over the output of the encoder stack.
- employ a residual connection around each of the two sub-layers, followed by layer normalization.
- modify the self-attention sub-layer, prevent positions from attending to subsequent positions.

解码器:

- 6个相同的层，每层除了有与**编码器**相同的2个子层之外，第3个层对**编码器**层的输出执行多头注意力操作。
- 两个子层的每个子层周围采用残差连接，并进行归一化。
- 修改了自我关注子层，防止位置关注到后续位置。

3.2. Attention

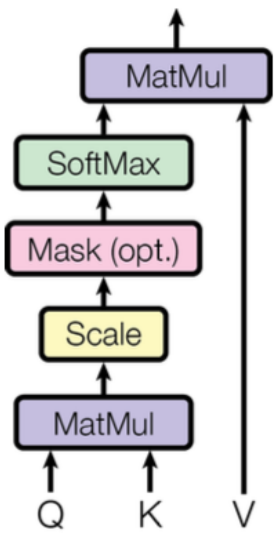
An **attention function** can be described as **mapping** a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

注意力函数可以描述为将一个查询和一组键值对**映射**到一个输出，其中查询、键、值和输出都是向量。

输出是以值的加权和来计算的，其中分配给每个值的权重是通过查询与相应键的兼容函数来计算的。

3.2.1. Scaled Dot-Product Attention



- Input **queries**, **keys**, **values**. the dimension of **queries** and **keys** are d_k, and **values** are d_v.
- 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**.

- compute the attention function on a set of **queries** simultaneously, compute **queries** matrix Q, **keys** matrix K and **values** matrix V as :$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$
- 输入**查询**、**键**、**值**，**查询**和**键**的维度为d_k，**值**的维度为d_v。
- 计算**查询**与所有**键**的点积，将每个点积除以 $\sqrt{d_k}$，然后应用**softmax函数**获取**值**的权重。
- 同时对一组**查询**计算注意力函数，**查询**矩阵Q、**键**矩阵K以及**值**矩阵V的计算公式为：$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$

The two most commonly used attention functions are **additive attention**, and **dot-product (multiplicative) attention**.

Additive attention computes the compatibility function using a feed-forward network with a single hidden layer.

Dot-product attention is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

最常用的两种注意力函数是**加法注意力**和**点积（乘法）注意力**。

加法注意力使用单隐层的前馈网络计算相容函数。

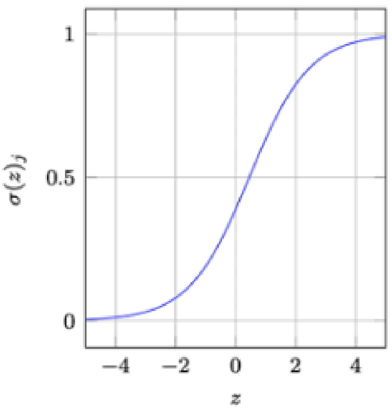
点积注意力在实践中速度更快，空间效率更高，因为它可以使用高度优化的矩阵乘法代码来实现。

Additive attention outperforms **dot-product attention without scaling** for larger values of d_k.

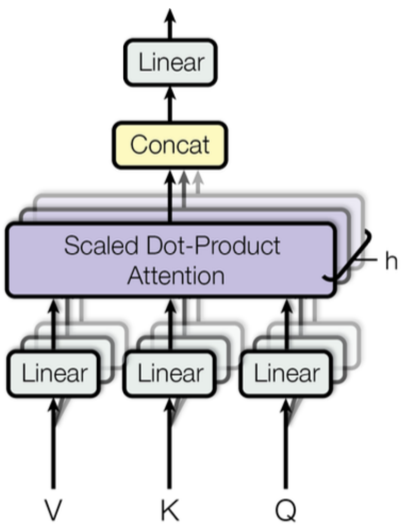
We suspect that for large values of d_k, the **dot-products** grow large in magnitude, pushing the **softmax** function into regions where it has extremely small gradients, so need scale the **dot-products** by $\frac{1}{\sqrt{d_k}}$

在d_k值较大的情况下，**加法注意力**在不缩放的情况下优于**点积注意力**。

我们怀疑，对于较大的d_k值，点积的幅度会越来越大，从而将**softmax**函数推向梯度极小的区域。为了消除这种影响，我们将点乘缩放$\frac{1}{\sqrt{d_k}}$ 。



3.2.2. Multi-Head Attention



- Linearly project the **queries**, **keys** and **values h times** with different, learned linear projections to d_k, d_k and d_v dimensions, respectively.

  将**查询**，**键**和**值**通过不同的、学习过的线性投影分别线性投影到d_k、d_k、d_v维度上**h次**。

- On each of these projected versions of **queries**, **keys** and **values** we then perform the **attention function** in parallel, yielding d_v dimensional output values.

  对每个投影版本的**查询**、**键**和**值**并行执行**注意力函数**，产生d_v维输出值。

- These are concatenated and once again projected, resulting in the final **values**.

  这些值被串联起来并再次投影，从而得到**最终值**。

Multi-head attention allows the model to **jointly attend** to information from different representation subspaces at different positions.

With a single attention head, **averaging inhibits this**.

多头注意力允许模型在不同位置**共同关注**来自不同表征子空间的信息。

而在单注意头的情况下，**平均化会抑制这一点**。

$$MultiHead(Q,K,V) = Concat(H_1,...,H_h)W^O$$

$$where H_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$W_i^Q \in \mathbb{R}^{d_{model}*d_k}, W_i^K \in \mathbb{R}^{d_{model}*d_k}, W_i^V \in \mathbb{R}^{d_{model}*d_v}$$

$$W^O \in \mathbb{R}^{hd_v*d_{model}}$$

- In this work, $h = 8$
- $d_k = d_v = \frac{d_{model}}{h} = 64$
- The total computational cost is similar to that of **single-head attention** with full dimensionality due to the **reduced dimension of each head**. $64 * 8 = 512 \approx 512$

3.2.3. Applications of Attention in our Model

Three different ways:

1. **In "encoder-decoder attention" layers**, the **queries** come from the **previous decoder layer**, and the memory **keys** and **values** come from **the output of the encoder**. This allows **every position in the decoder** to attend over all positions in the input sequence.

   在 "编码器-解码器注意 "层中，**查询**来自**前一个解码器层**，而**记忆键**和**记忆值**则来自**编码器的输出**。这使得**解码器中的每个位置**都能关注输入序列中的所有位置。

2. **The encoder contains self-attention layers.** In a self-attention layer all of the **keys**, **values** and **queries** come from the same place(the output of the previous layer in the encoder). Each position in the encoder can attend to all positions in the previous layer of the encoder.

   编码器包含自注意层。在自注意层中，所有的**键**、**值**和**查询**都来自同一个地方，即编码器中上一层的输出。编码器中的每个位置都可以关注编码器上一层的所有位置。

3. **Self-attention layers in the decoder** allow each position in the decoder to attend to all positions in the decoder up to and including that position. We need to **prevent leftward**

information flow in the decoder to **preserve the auto-regressive property**. We implement this inside of scaled dot-product attention by masking out (setting to $-\infty$) all **values** in the input of the **softmax** which correspond to **illegal connections**.

解码器中的自关注层允许解码器中的每个位置关注解码器中包括该位置在内的所有位置。我们需要**防止解码器中的信息向左流动**，以**保持自动回归特性**。我们通过屏蔽（设置为$-\infty$）**softmax** 输入中所有与非法连接相对应的值。

3.3. Position-wise Feed-Forward Networks

This network is applied to each position separately and identically, consists of two linear transformations with a ReLU activation in between.

该网络分别对每个位置进行相同的处理，由两个线性变换以及中间的一个 ReLU激活函数构成。

- $FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$
- As same as through 2 convolutions with kernel size 1.
- **dimensionality**: $d_{input} = 512 \rightarrow d_{middle} = 2048 \rightarrow d_{output} = 512$

3.4. Embeddings and Softmax

In our model, we share the same **weight matrix** between the **two embedding layers** and the **pre-softmax linear transformation**.

在我们的模型中，我们在**两个嵌入层**和**pre-softmax线性变换**之间共享相同的**权重矩阵**。

3.5. Positional Encoding

In order for the model to make use of the order of the sequence.

为了让模型能够利用序列的顺序。

We add "**positional encodings**" to the **input embeddings** at the bottoms of the encoder and decoder stacks.

我们在编码器和解码器堆栈底部的**输入嵌入**中添加了 "**位置编码**"。

The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed.

There are many choices of positional encodings, **learned** and **fixed**(two versions produced nearly identical results).

位置编码的维度d_{model}与嵌入式编码相同，因此两者可以相加。

位置编码有多种选择，包括**学习编码**和**固定编码**(二者产生的结果几乎相同)。

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

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

- pos is position; i is the dimension;

  pos是位置；i是维度

- each dimension of the positional encoding corresponds to a sinusoid.

  位置编码的每个维度对应一个正弦波。

- any fixed offset k, $PE(pos+k)$ can be represented as a linear function of $PE(pos)$.

  对于任何固定的偏移 k，$PE(pos+k)$都可以表示为$PE(pos)$的线性函数。

We chose the **sinusoidal version** because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

我们之所以选择**正弦波版本**，是因为它可以让模型推断出比训练时遇到的序列长度更长的序列。

4. Why Self-Attention

- **total computational complexity**

  总体计算复杂度

- **the amount of computation that can be parallelized**

可并行化的计算量

- **the path length between long-range dependencies in the network**

  网络中长距离依赖关系之间的路径长度。

Learning long-range dependencies is a key challenge in many sequence transduction tasks.

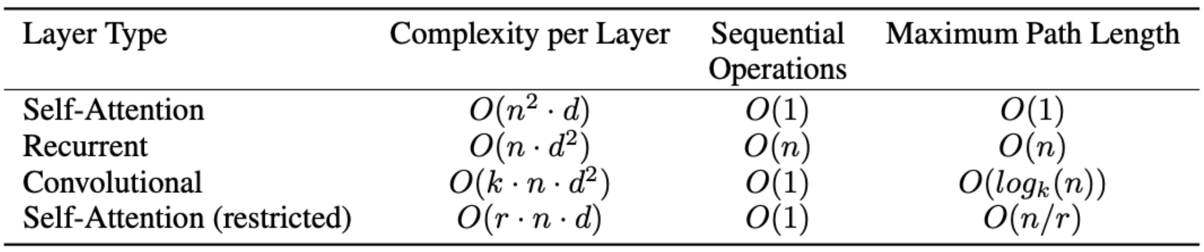
学习长程依赖关系是许多序列转导任务的关键挑战。

One key factor affecting the ability to learn such dependencies is **the length of the paths** forward and backward signals have to traverse in the network.

影响学习此类依赖关系能力的一个关键因素是前向和后向信号在网络中必须穿越的**路径长度**。

The shorter these **paths between any combination of positions in the input and output** sequences, the easier it is to learn long-range dependencies.

输入和输出序列中任意位置组合之间的路径越短，学习远距离依赖关系就越容易。



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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

- n is the **sequence length**

  n 为**序列长度**

- d is the **representation dimension**

  d 为**表征维度**

- k is the **kernel size of convolutions**

  k为**卷积核尺寸**

- r is the **size of the neighborhood in restricted self-attention**

  r为**受限自注意力临域大小**

In terms of **computational complexity**, self-attention layers are faster than recurrent layers when the **sequence length n is smaller than the representation dimensionality d**.

就**计算复杂度**而言，当序列长度 n 小于表示维度 d 时，自注意层的计算速度要快于递归层。

To improve computational performance for tasks involving **very long sequences**, self-attention could be restricted to considering **only a neighborhood of size r in the input sequence centered around the respective output position**.

为了提高涉及**超长序列**任务的计算性能，可以限制自我关注**只考虑输入序列中以相应输出位置为中心的大小为 r 的邻域**。

5. Training

5.1. Training Data and Batching

5.2. Hardware and Schedule

5.3. Optimizer

Varied the learning rate follow:

$$learn\_rate = d_{model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

This corresponds to **increasing the learning rate linearly** for the first **warmup_steps** training steps, and **decreasing it thereafter proportionally** to the inverse square root of the step number.

这相当于在第一个warmup_steps 训练步数中**线性增加**学习率，此后学习率**按步数的平方反比例递减**。

5.4. Regularization

Residual Dropout:

We apply **dropout** to the output of each sub-layer, before it is added to the sub-layer input and normalized. In addition, we apply **dropout** to the sums of the **embeddings** and the **positional encodings** in both the encoder and decoder stacks.

我们将**dropout**应用于每个子层的输出，然后将其添加到子层输入中并进行归一化处理。

此外，我们还对编码器和解码器堆栈中的**嵌入**和**位置编码**之和进行了dropout。

- $P_{drop} = 0.1$

Label Smoothing:

- $\varepsilon_{ls} = 0.1$

6. Result

6.1. Machine Translation

6.2. Model Variations

Single-head attention worse than the best setting, quality also drops off with **too many heads**.

单头注意力要比最好的设置性能稍差，但是注意力头过多也会导致性能的下降。

7. Conclusion

In this work, we presented the **Transformer**, the first sequence transduction model based **entirely on attention**, replacing the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention.

在这项工作中，我们提出了 "**Transformer**"，这是第一个**完全基于注意力**的序列转换模型，用多头自我注意力取代了编码器-解码器架构中最常用的递归层。