# Deep Learning Model for NLP Task - July 5'th, 2019

Transformer Model

申恒恒 - https://github.com/rh01

## MyToc



## Transformer 背景知识

Transformer 是 Google 在 2017 年做机器翻译任务时，Vaswani et al., 2017 在论文 **Attention is all you need** 中提出的，当时引起很大的反响，每一位做 NLP 都必须要知道 Transformer，十分重要。

不过这里没打算重点介绍它，想要入门 Transformer 的可以参考以下三篇文章：一个是 Jay Alammar 可视化地介绍 Transformer 的博客文章 The Illustrated Transformer，非常容易理解整个机制，建议先从这篇看起，这是中文翻译 版本；第二篇是 Calvo 的博客：Dissecting BERT Part 1: The Encoder ，尽管说是解析 Bert，但是因为 **Bert 的 Encoder 就是 Transformer**，所以其实它是在解析 Transformer，里面举的例子很好；再然后可以进阶一下，参考哈佛大学 NLP 研究组写的 The Annotated Transformer.，代码原理双管齐下，讲得也很清楚。

Attention 主要用来帮助提升 NMT（Neural Machine Translation）模型的翻译效果的模型，而 Transformer 则是用来加速 NMT 训练的一个模型，该模型不仅扩展了 Attention 而且在 Google 的 NMT 中表现突出，他

改变了传统的 NLP 模型不能并行的通病，下面主要借鉴 The Illustrated Transformer 文章来对 Transformer 进行深度解析。

**Transformer 结构**

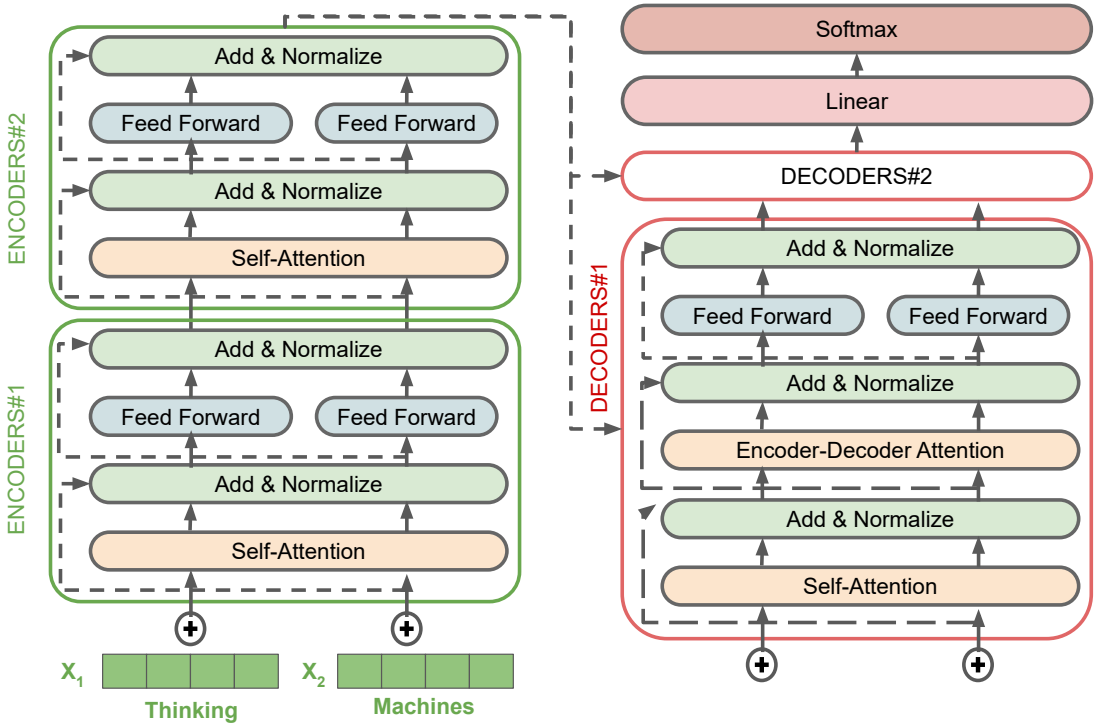


图 1：两层编码器 + 两层解码器组成 Transformer

source url of image: `https://jalammar.github.io/illustrated-transformer/`

其中，在编码器结构中，每个子层中（Slef-Attention, Feed Forward NN），都有残差连接，并且紧跟着 Layer-Normalization。

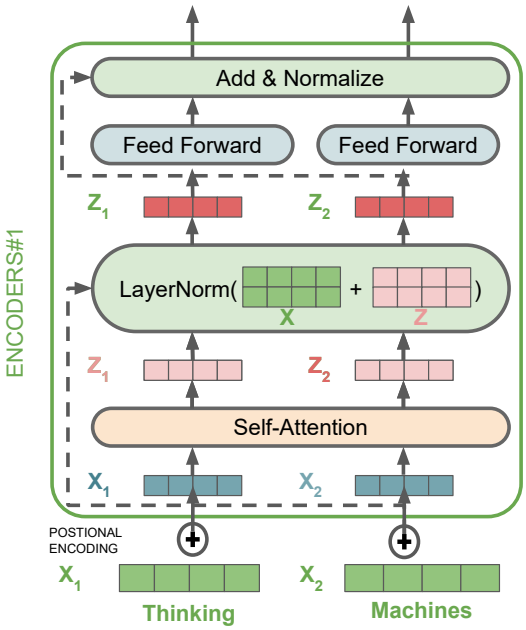


图 2：Transformer 中的编码器结构向量版

source url of image: `https://jalammar.github.io/illustrated-transformer/`

最后这里给出Vaswani et al., 2017 论文的 Transformer 架构图：

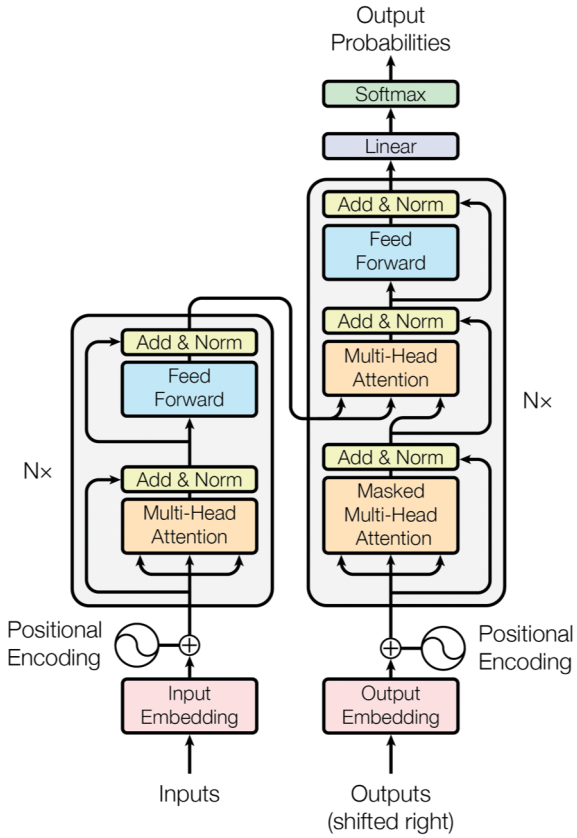


图 3：The Transformer - model architecture. (Fig 1. Vaswani et al., 2017 )

**Transformer 算法流程**

- Scaled Dot-Product Attention
- Multi-Head Attention
- Position-wise Feed-Forward Networks
- Embeddings and Softmax
- Positional Encoding

**Scaled Dot-Product Attention**

Q: queries, K: keys, V: values

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

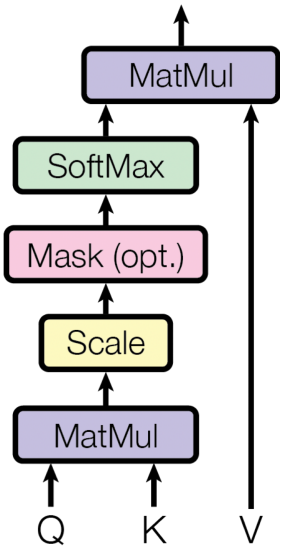


图 3：Scaled Dot-Product Attention (Fig 2(left). Vaswani et al., 2017 )

**Multi-Head Attention**

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head } 1, \ldots, \text{head } h)W^O$$
$$\text{where head } i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ .

The total computational cost is similar to that of single-head attention with full dimensionality

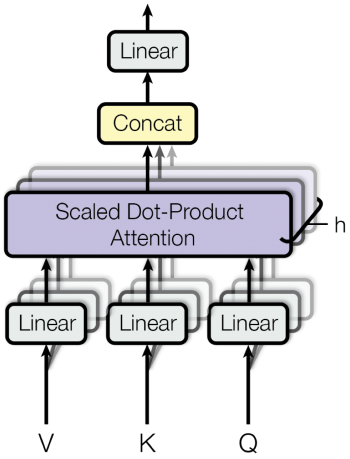


图 4：Multi-Head Attention (Fig 2(right). Vaswani et al., 2017 )

**Position-wise Feed-Forward Networks**

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

**Embeddings and Softmax**

Share embedding weights and the pre-softmax linear transformation (refer to arXiv:1608.05859)

**Positional Encoding**

**Reason.** no RNN to model the sequence position

**Two types.**
- learned positional embeddings (arXiv:1705.03122v2)
- Sinusoid:

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