Transformer 模型的处理流程

1. 输入预处理

首先，将输入的文本序列进行分词处理，转化为一系列 token（词语或子词），这一步可以默认是对单词进行的分割。对每个 token 应用嵌入层，对于每个单词查询对应的向量，将其转化为低维稠密向量表示，同时加上位置编码，以便模型能够区分不同位置的 token。

2. 编码器（Encoder）阶段

编码器的核心是对输入序列执行**多头自注意力机制**。每个自注意力头部独立计算输入序列中各个位置之间的关联性，最后将各个头部的结果合并，得到一个新的上下文相关的向量表示。这一过程允许模型同时关注序列中的所有位置，并动态调整各位置的相对重要性。

$$S = \frac{QK^T}{\sqrt{d_k}}$$

$$A = \text{softmax}\,(S) = \text{softmax}\,(\frac{QK^T}{\sqrt{d_k}}) = [\frac{\exp(s_{ij})}{\sum_{k=1}^{n_k} \exp\,(s_{ik})}]$$

$$\text{AttentionOutput} = AV$$

在 Transformer 架构中，Q、K、V 是自注意力机制中的核心组成部分，分别代表查询（Query）、键（Key）和值（Value）矩阵。Query 矩阵是从当前上下文中通过线性变换得到的。对于每个位置的输入向量，模型会通过一个特定的权重矩阵（Wq）对其进行变换，生成对应的查询向量，这个查询向量用来衡量该位

置与其他所有位置之间的相关性；Key 矩阵也是由输入序列的每个位置通过不同的线性变换产生的，同样地，每一个位置的输入向量通过另一个权重矩阵（Wk）得到键向量。键向量主要用于和查询向量进行匹配计算，以确定哪些位置的输入信息对当前位置最重要；Value 矩阵对应的是存储有用信息的矩阵。输入序列的每个位置的向量通过第三个权重矩阵（Wv）转换成值向量，这些值向量包含了原始输入的特征信息，但被重新加权以突出那些与当前查询向量最相关的部分。

在自注意力机制中，首先计算 Query 和 Key 的点积，用于评估两个向量的相关程度，然后除以系数处理后通过 softmax 函数归一化，得到注意力权重分布。接下来，将这个注意力权重分布应用于 Value 矩阵，通过对 Value 矩阵按权重求和，生成每个位置的上下文向量。

以上的步骤可被视为单头注意力机制，即通过一次查询和接下来的比较来判断各方向的相似性。多头注意力进一步扩展了这一概念，它让模型同时从多个不同的角度（或者说"头"）关注输入序列的不同方面。每个注意力头都有自己独立的 Q、K、V 矩阵，从而学习到不同类型的依赖关系。最后，来自各个头的输出会通过线性投影后拼接，再次经过线性变换，产生最终的注意力输出。这样设计增强了模型捕捉复杂模式的能力，并增加了模型的表达力。

自注意力层的输出会紧接着进入一个前馈神经网络，通常包含两个线性层，中间可能会有一个激活函数（如 ReLU）提供非线性变换能力。FFN 的输出又会通过残差连接与原始自注意力层的输出相加，之后再经过层归一化稳定训练过程和提高模型表现。

上述编码器层会被堆叠多层，形成编码器栈，每一层都会对上一层的输出进行同样的处理，从而逐步提取出更高层次的抽象信息。

3.解码器（Decoder）阶段

不同于编码器，解码器在执行自注意力时会对未来的 tokens 位置进行遮蔽，确保在预测当前时刻的输出时不依赖未来时刻的信息。解码器会根据当前解码器的状态与编码器的输出进行注意力计算，获取源序列的上下文信息。与编码器类似，解码器内部同样包括前馈神经网络、残差连接和层规范化操作。经过多层解码器处理后，最后一个解码器层的输出会进入一个线性层映射到词汇表大小的空间，并通过 softmax 函数转换为概率分布，代表每个词汇作为下一个生成词的概率。