Transformer技术原理

论文原文与对照翻译 transformer翻译.pdf

概述

Transformer的核心是一个编码器-解码器架构。它通过自注意力机制 来工作：编码器首先并行处理整个输入序列，通过多头自注意力让每个词都能关注到序列中所有其他词，生成富含上下文信息的表示；解码器则利用编码器的输出和已生成的目标序列，通过掩码自注意力和编码器-解码器注意力来逐个生成目标词。其关键在于完全依赖注意力机制进行全局信息关联，摒弃了循环和卷积结构，从而实现了高效的并行计算，并有效解决了长距离依赖问题。

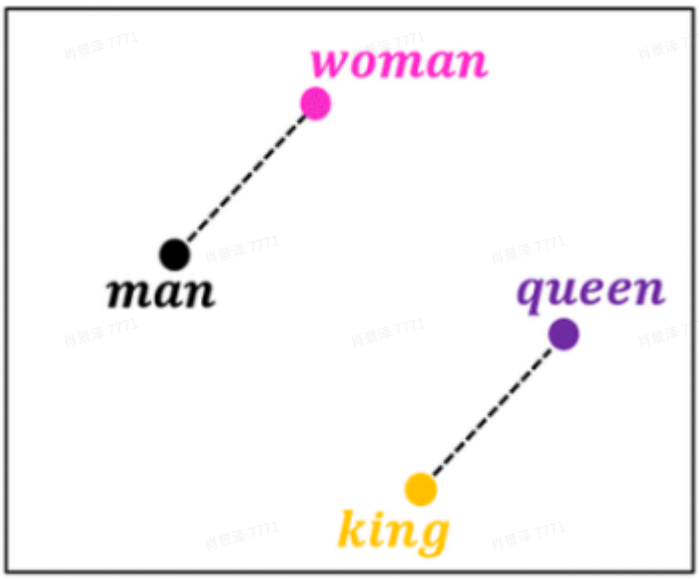
原理

编码器

输入 → 多头注意力 → 残差连接&层归一化 → 前馈神经网络 → 残差连接&层归一化 → 输出

词嵌入

奠基方法：word2vex

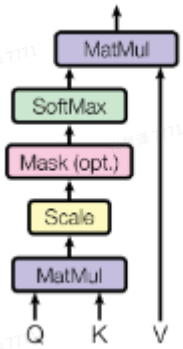


将词语字典构建于一个向量空间中，语义邻近的词元会在相近的空间位置中。这种方式可以帮助模型更好地理解词语直接的关系。

自注意力机制

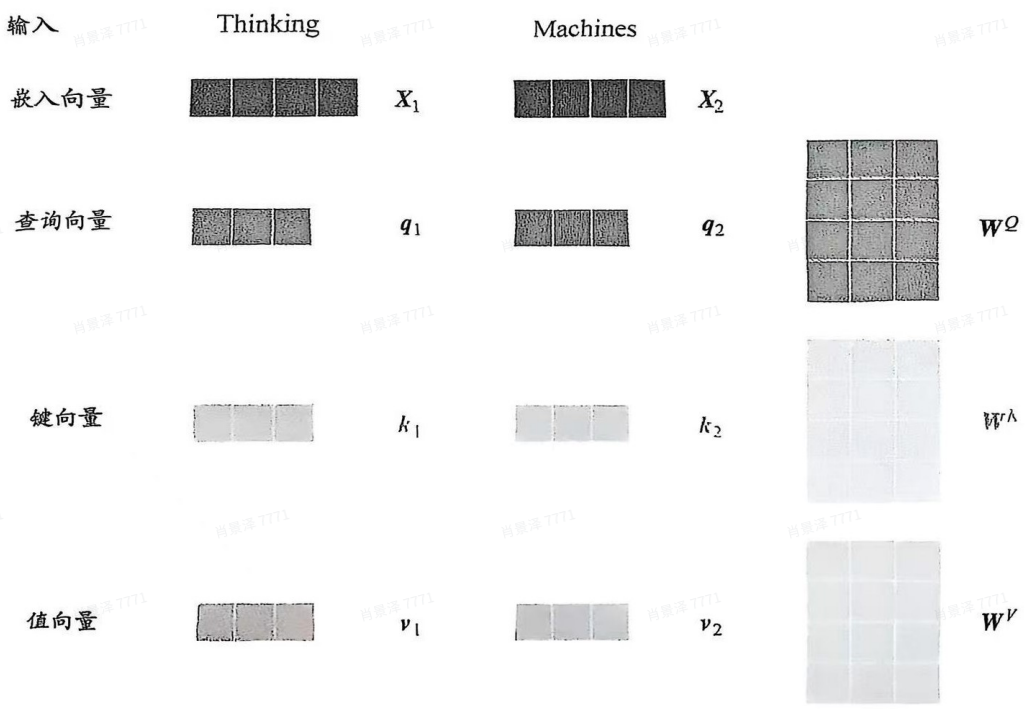
一个注意力函数可以描述为将查询和一组键值对映射到输出，其中查询q、键k、值v和输出都是向量。输出计算为加权求和。

缩放点积注意力

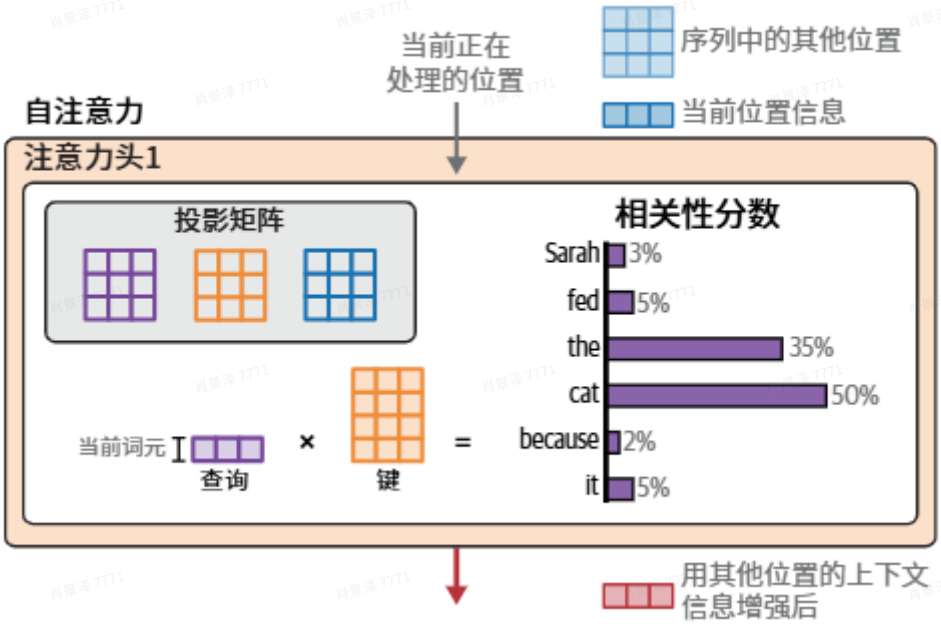


Transformer对输出文本的每个词元进行向量表征，向量表征由词向量和其对应位置的编码向量相加构成。

为每个词元构建q、k、v这三个向量，这三个向量由词向量和初始化矩阵Wq、Wk、Wv相乘而得（这三个矩阵的参数在开始时随机初始化，训练过程中通过梯度下降不断调整）。



然后为每个词元进行缩放点积注意力运算，得到注意力分数。



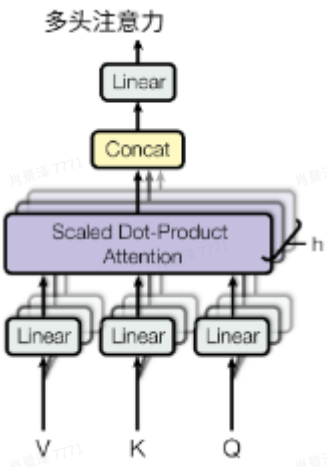
将这个词元的q向量和每个词元的k向量的点积后除以归一化常数d，得到softmax权重系数，再将此系数与对应的v向量相乘后求和，得到单个词元的加权v。即当前位置词元的最终向量表示z。

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

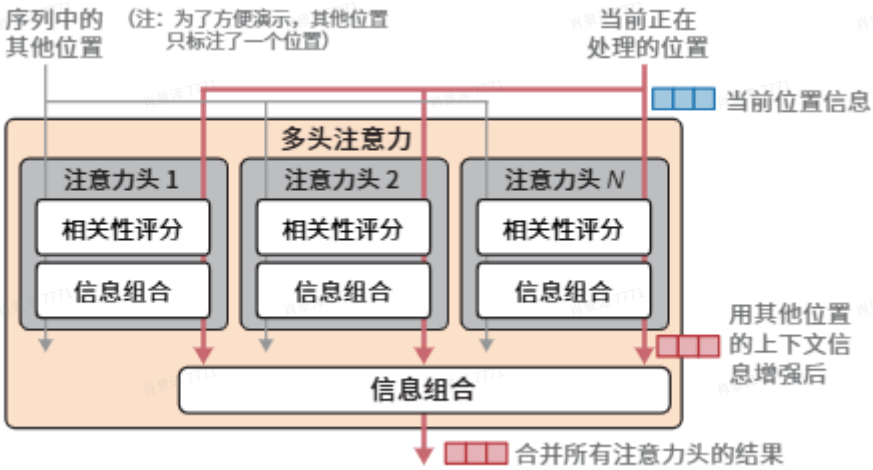
并行词元处理

为提高计算速度，transformer模型使用矩阵运算来实现上述流程。即最终得到的z是包含所有词元最终向量的向量集合[z1,z2,z3,...]。

多头注意力机制



在深度学习过程中，几乎不存在唯一的“全局最优解”。对于不同的初始化矩阵Wq、Wk、Wv，在应用于同一个输入中，梯度下降会收敛到不同的最优解，即全局最优的等价解。所以多个不同初始化的注意力头可以捕捉到不同的特征，而这些特征皆是最有用的，即不同空间区域的最优解。



Transformer 模型中使用了8个注意力头来提升自注意力的表现，即在解码器和编码器中各配了8个矩阵集合。首先把词向量降维投影到低维子空间中，经过8个注意力头得到的8个低维矩阵，将这些低维矩阵直接拼接后使用一个额外的权重矩阵与拼接矩阵相乘，得到前馈神经网络层的输入。

残差连接与层归一化

残差连接为了更方便地保留原始信息，第一层网络中多头注意力的输出会加上原始输入得到x+z。残差连接通过保留原始输入，为梯度反向传播提供了直接通路，有效解决了深度网络中的梯度消失问题，使得训练极深网络成为可能。

然后对x+z层归一化LayerNorm（x+z）为正态分布，来避免训练过程中的梯度爆炸，提升训练稳定性。层归一化相当于对每次训练环境进行控制变量，防止绝对值差距过大带来的误差，排除分布变化干扰，专注学习特征表示。

前馈神经网络层

前馈神经网络包含了两个线性变化层，中间使用ReLU激活函数，即ReLU（x）= max（0，x）。

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

上式中W1为升维矩阵，W2为降维矩阵，将原始向量展开到高维空间可以发现更多细节特征。

ReLU激活函数则可以实现以下作用：

- 关闭不重要的特征通路（负值→0）
- 保留重要的特征通路（正值不变）
- 创建稀疏表示

位置编码

Transformer的自注意力机制是“无序的”。它同时处理所有单词，并通过加权和来计算关系。所以Transformer模型本身无法感知单词顺序。论文里采用的位置编码，它不是为每个位置分配一个独立的

编号，而是用一系列不同频率的“波”来共同表示一个位置，这样模型就能轻松理解位置之间的“相对关系。

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

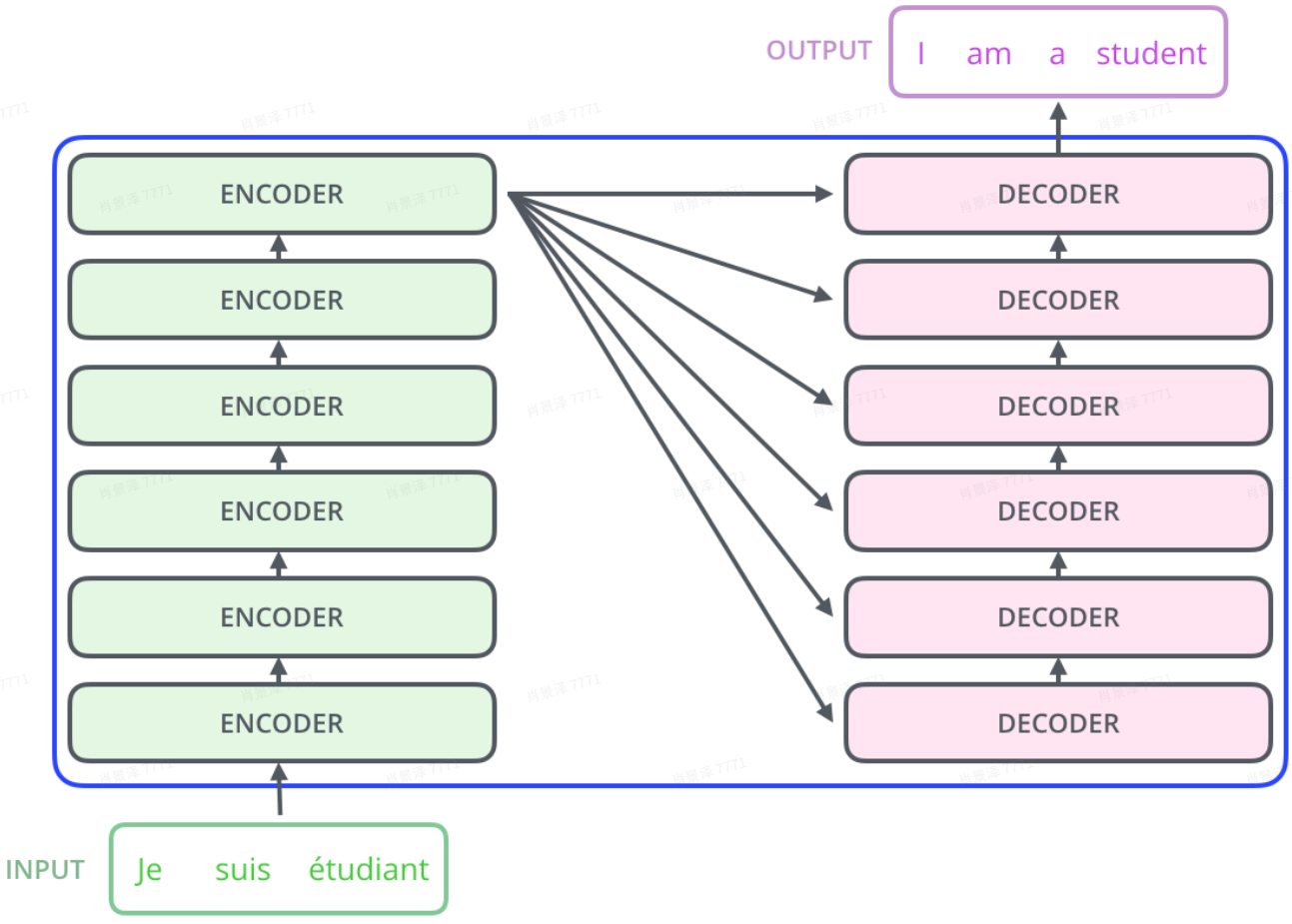
- `pos`：单词在句子中的绝对位置。比如第一个词 `pos=1`，第二个词 `pos=2`。
- `i`：位置编码向量的维度索引。假设位置编码向量是4维的（`d_model=4`），那么 `i` 可以是 0, 1（因为 `2i` 和 `2i+1` 会覆盖所有维度）。
- `d_model`：词向量的总维度（在原始论文中是512）。

位置编码和词向量相加后，模型在经过训练可以轻易地分离出来两者。因为词向量所在的高维空间提供了巨大的“容量”，词义混淆的概率极低。位置编码的扰动相对较小，不会覆盖主要的词义信息。

解码器

输入（已生成的序列，右移） → Masked多头自注意力 → 残差连接&层归一化 → 编码器-解码器注意力（交叉注意力） → 残差连接&层归一化 → 前馈神经网络 → 残差连接&层归一化 → 输出

Transformer的模型架构如下图



解码器工作原理和编码器相似，但比编码器多出一层交叉注意力层用于接收编码器最顶层输出，编码器最顶层的输出为解码器的多头注意力计算提供K矩阵和V矩阵，然后在将编码器的输出和本次解码器的输出通过前向传播输出给下一个解码器。尽管多头注意力机制能捕捉很多特征，但需经过多层神经网络的提炼和抽象，以让模型能理解复杂的任务。

同时，解码器的顶部输出会经过一个全连接层，将特征维度映射到词表大小，然后经过softmax运算得到词表中每一个词的概率，选择概率最大的值作为解码出来的词元。

对于每一层编码器和解码器的内部结构如下图：

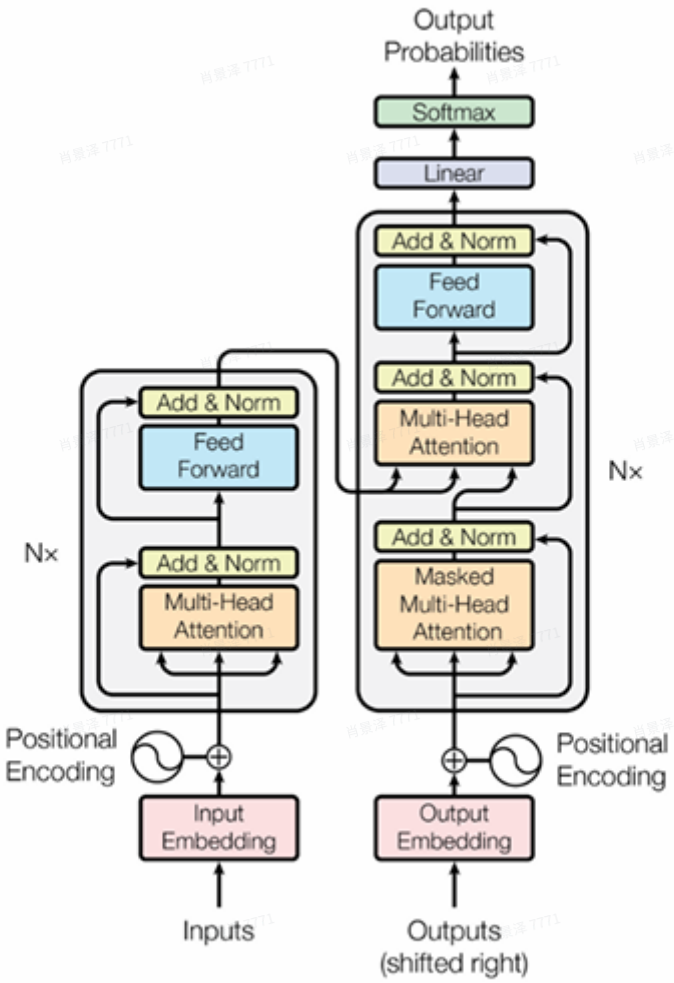


Figure 1: The Transformer - model architecture.

在训练中，每层解码器的输入采用shifted right，确保输入和输出在位置上正确对齐，为模型提供完美的“上一个词”作为提示。在自注意力计算时，掩码会遮挡住当前词之后的所有词。比如解码器的输入是 `Je t' aime`，对于输入序列中的第三个词 “t ‘” ，掩码会阻止它看到它后面的 “aime”。因此，模型在计算 “t’ ” 的表示时，只能用到 `Je` 和它自身 `t‘` 的信息。

自注意力机制和RNN、CNN对比

对比维度	自注意力层 (Self-Attention)	循环层 (Recurrent)	卷积层 (Convolutional)
每层复杂度	$O(n^2 \cdot d)$	$O(n \cdot d^2)$	$O(k \cdot n \cdot d)$
顺序操作数	$O(1)$	$O(n)$	$O(1)$
最大路径长度	$O(1)$	$O(n)$	$O(\log_k(n))$ 或 $O(n/k)$

特性	自注意力 (Transformer)	循环神经网络 (RNN/LSTM)	卷积神经网络 (CNN)
核心原理	**全局并行关联**：通过查询、键、值机制计算序列中所有位置间的全局依赖关系。	**顺序状态传递**：按时间步逐步处理，隐藏状态作为记忆单元传递历史信息。	**局部特征提取**：使用卷积核在序列上滑动，通过局部连接和权重共享提取特征。
处理序列方式	完全并行。一次性看到整个序列，计算所有词对之间的关系。	严格串行。必须依次处理每个时间步，当前状态依赖于前一状态。	局部并行。卷积核在序列上滑动计算，但每个窗口内可并行。
长距离依赖处理	**最优**。任意两位置间路径长度为O(1)，直接连接，能完美捕捉长距离依赖。	**差**。路径长度为O(n)，存在梯度消失/爆炸问题，难以学习长程依赖。	**中等**。路径长度为 $O(\log_k(n))$，通过堆叠深层网络扩大感受野来捕捉。
计算复杂度	$O(n^2 \cdot d)$，n为序列长度，d为特征维度。序列较长时计算量显著增加。	$O(n \cdot d^2)$。串行计算导致训练慢，但推理时复杂度较低。	$O(k \cdot n \cdot d)$，k为卷积核大小。通常k<<n，计算效率较高。
并行化能力	**极高**。所有位置的计算可完全同时进行，非常适合GPU加速。	**极低**。本质串行，无法利用现代硬件的并行能力，训练速度慢。	**高**。卷积核在不同位置的运算是独立的，可高度并行化。
路径长度（关键指标）	$O(1)$ - 常数级别。前向/反向传播信号一步直达任意位置。	$O(n)$ - 线性级别。信号需要一步步传递，路径很长。	$O(\log_k(n))$ - 对数级别。通过多层叠加，路径显著短于RNN。
优势	1. 强大的长程建模能力 2. 高度并行，训练速度快	1. 天然适合时序数据 2. 理论上的无	1. 局部特征提取能力强 2. 平移不变性

优势	训练速度快 3. 模型可解释性较好（注意力权重）	限记忆能力 3. 推理时计算效率高	3. 训练稳定，对超参数不敏感
劣势	1. 计算复杂度随序列长度平方增长 2. 位置信息需要显式编码 3. 大量数据下才能发挥优势	1. 并行性差，训练慢 2. 长程依赖学习困难 3. 梯度消失/爆炸问题	1. 捕捉全局依赖需要深层网络 2. 对序列顺序不敏感（除非使用因果卷积） 3. 固定卷积核可能不够灵活
主要应用场景	机器翻译、文本生成、BERT/GPT等大语言模型、语音识别。	时间序列预测、情感分析、简单语言模型、股票预测。	文本分类、情感分析、机器翻译（早期）、语音合成（WaveNet）。
直观比喻	**圆桌会议**：每个人同时与所有其他人交流，快速形成全局共识。	**传话游戏**：信息从第一个人依次传到最后一个人，容易失真或遗忘。	**显微镜扫描**：用固定模式的镜头逐区域观察，再拼凑整体画面。