**学习进度**

本周学习任务：

学习李宏毅老师，关于注意力机制及Transformer的视频课程；

学习李沐老师《动手学深度学习（PyTorch版）》，关于预备知识部分的内容。

1. 关于注意力机制的补充学习

为了更好地适应变长输入情景，**注意力机制**通过给定上下文来计算注意力权重的方式，捕捉了不同位置之间的重要关联信息，从而实现了原始输入序列分量的重构。值得注意的是，给定的上下文信息并不一定泛指全局上下文。此外，注意力机制通过矩阵计算的方式，能够同时重构所有的序列分量，显著提高了并行性。

注意力机制的主要类型有：自注意力、多头注意力、引入位置编码的自注意力及截断式自注意力机制。在**多头注意力机制**中，每个输入序列分量可与多个q、k、v对应，不同的q表达了不同类型的相关性。

此外，**自注意力机制**只关注分词之间的语义关系，从而忽略了分词的位置信息。具体而言，无论输入序列中的元素如何排列，自注意力机制都会产生相同的注意力权重。优点是：在翻译任务中，由于输入输出序列词序的不同，自注意力机制能够帮助模型在不受位置信息限制的情况下，有效捕捉分词之间的语义关系，从而提高了翻译模型的适用性及泛化能力。缺点是：在诸如词序标注等任务情景中，忽略位置信息，往往是不可容忍的。如图1.1所示，**引入位置编码的自注意力机制**，通过添加位置向量的方式（每个位置上的分量都有唯一的位置向量相对应），完成了位置信息的记录。此外，获取位置向量的方式主要有：Rule-based hand crafted、learned from data。

graphic

图1.1 位置编码的导入

显而易见的是，注意力矩阵的规模等于输入序列长度的平方。然而，在诸如音频处理等任务中，由于输入序列太长，往往极大影响计算效率。为此，**截断式自注意力机制（Truncated Self-attention）**，通过分步提取局部上下文的方式，缓解了序列太长导致的运算过慢问题。

1. 注意力机制与传统模型的关系

**循环神经网络RNN与注意力机制**，都能有效考虑到输入序列中的上下文信息。然而，相较于RNN，注意力机制无需等待前一时间步的输出，就可以同时计算出不同步的结果。此外，相较于注意力机制，RNN还存在长期记忆遗失问题。

对于**卷积神经网络CNN**而言，CNN可以理解为只考虑局部上下文的自注意力机制。相较于其它传统模型，CNN能够灵活地运用注意力机制改造自身结构，比如：引入通道注意力模块，给予通道不同的关注度；引入空间注意力模块，给予特征点不同的关注度。

此外，如图2.1所示，引入自注意力机制的图（Graph）模型，能够考虑到不同边（edge）的重要程度，这也是一种常见的图神经网络类型。

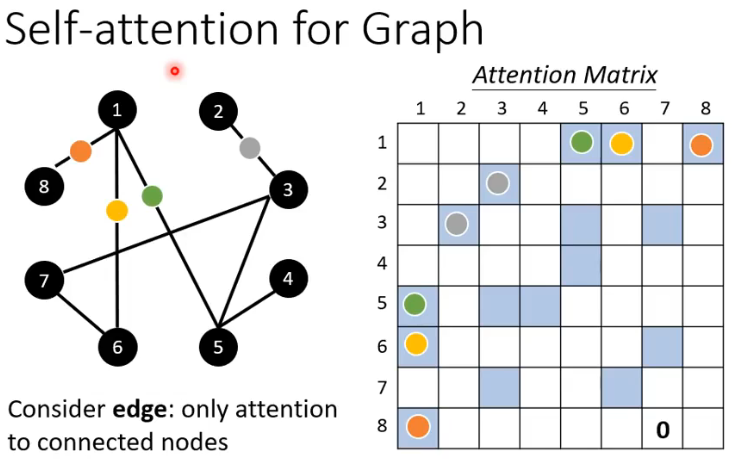


图2.1 引入自注意力机制的图模型

1. 基于注意力机制的Transformer

（1）网络架构

如图3.1所示，Transformer模型由左侧的Encoder block、右侧的Decoder block构成。Encoder block包含有：Multi-Head Attention+Add&Norm、Feed Forward+Add&Norm，Decoder block包含有：**Masked** Multi-Head Attention+Add&Norm、Multi-Head Attention+Add&Norm及Feed Forward+Add&Norm。其中，**Add**表示残差连接（Residual Connection），用于防止网络退化；**Norm**表示Layer Normalization，用于对每层的激活值进行归一化；**Feed Forward**包含两层全连接层，第一层的激活函数为ReLu，第二层不使用激活函数。

graphic

图3.1 Transformer网络架构

1. 工作流程

在Encoder block中：首先，通过词嵌入（Embedding）方法将原始输入序列转化为一组向量，并分别加上对应的位置编码（Positional Encoding），用于引入分量的位置信息。然后，通过Multi-Head Attention+Add&Norm模块，捕捉不同位置之间的语义关联信息，并保证训练的稳定性。最后，通过Feed Forward+Add&Norm模块完成进一步的非线性变换，捕捉更复杂的语义特征及关联，从而获得分步考虑上下文及位置信息的编码信息矩阵C。

在Decoder block中：首先，通过Masked Multi-Head Attention+Add&Norm模块，分步接收当前预测词的前文输出序列，捕捉前文输出序列中不同位置之间的关联信息。然后，通过Multi-Head Attention+Add&Norm模块，接收前一模块的输出及来自Encoder block的编码信息矩阵C，分别计算多头注意力机制所需的Q及K、V。最后，通过Softmax获得当前步的预测分词，并拼接到前文输出序列中进行后续分词预测，直到生成完整的输出序列。

1. 关于测试的关键问题：“一步错，步步错”

在训练Transformer模型的过程中，Decoder block分步预测对应的输出分词时，Masked Multi-Head Attention+Add&Norm每次接收的前文输出序列是指：截取正确翻译结果（标签）的前文序列。然而，在测试Transformer模型的过程中，每次接收的前文输出序列是指：分步预测出的分词组合。显而易见的是，在测试阶段，如果前步预测出的分词是错误的，将导致后步预测出的分词更可能出现错误，从而发生“一步错，步步错”现象。

一种简单直接的做法是：在训练Transformer模型的过程中，Masked Multi-Head Attention+Add&Norm模块每次接收的前文输出序列，不应该是完全正确的答案。这使得训练出的模型，能够更好地应对测试中“一步错”的情景。

1. 可行的改进

值得注意的是，在Encoder部分可以堆叠多个Encoder block，甚至每个Encoder block中的模块都可以进行自由的重组，而在Decoder部分也同样如此。此外，如图3.2所示，如果存在多个Encoder block及Decoder block，每个Decoder block具体接收哪个Encoder block输出的编码信息矩阵C，也是一个十分自由的选择。

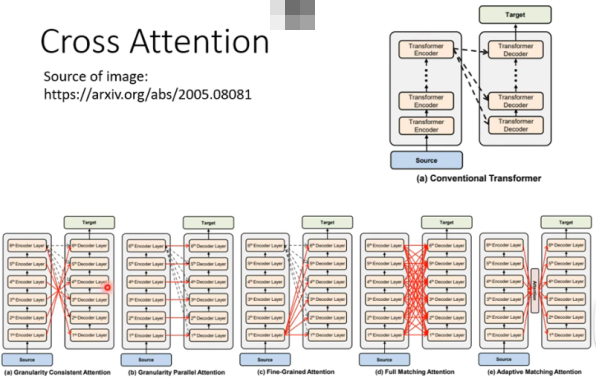


图3.2 Cross Attention

1. 学习《动手学深度学习（PyTorch版）》，关于预备知识部分的内容

- 下载最新的书籍配套代码，配置对应的虚拟环境，基于GPU测试运行了模型实例。

- 学习并简单实践了预备知识中：数据操作、线性代数、微分、自动求导及概率部分的内容。