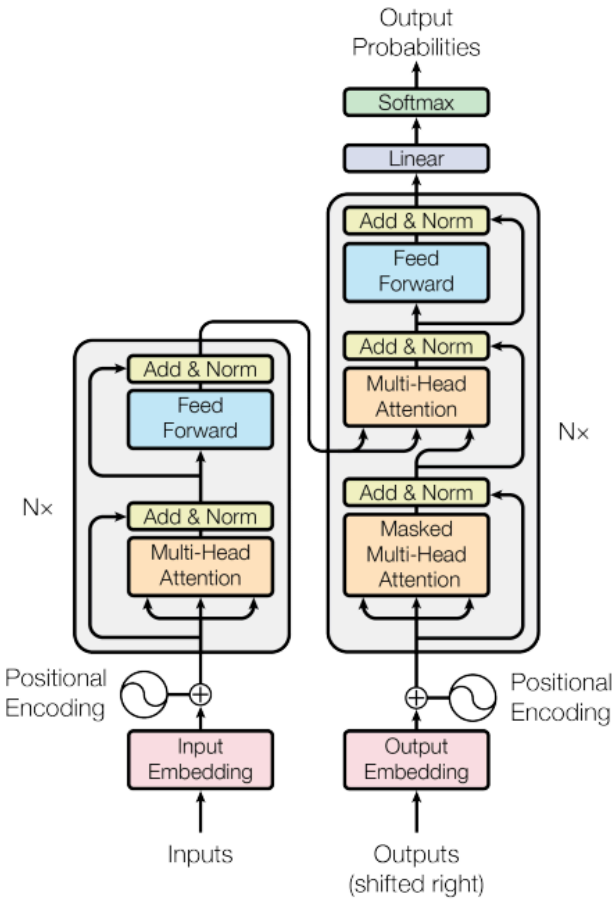
1. Transformer的结构



- Encoder （原文6个）
  - 多头self-attention模块
  - 点式前馈神经网络模块 FFN
- Decoder（原文6个）
  - 多头self-attention模块（需要mask）
  - **多头交互模块**（KV来自于Encoder的输出）
  - 点式前馈神经网络模块 FFN

每个子层后都跟有 add & layer norm层

2. FFN的作用

这层主要是提供非线性变换，增强模型表达能力。

另外，FFN linear 到4d，再linear回d（好像是个计算trick）

3. 为什么要multi-head attention?

- Attention is all you need论文中讲模型分为多个头，形成多个子空间，每个头关注不同方面的信息
- 多头的本质是多个独立的attention计算，然后进行集成，具有一定防治过拟合的作用

4. 为什么在进行多头注意力的时候需要对每个head进行降维?

保证concat后维度不变，转化为多个子空间丰富特征信息，同时能降低运算量。

5. Position Encoding

> 直接通过cos/sin计算而来，而非训练得到。

文本序列是有时序关系的，但是Attention无法捕获时序关系，所以需要添加位置信息进行补充学习。

位置信息需要满足如下两个要求：

- 每个位置有一个唯一的positional encoding.
- 两个位置之间的关系可以通过他们位置编码间的仿射变换来建模（获得）

Transformer用sin/cos设置position encoding：

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

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

比如，序列中第五个单词，其pos为4，它的position encoding为：

$$PE(0) = [sin(\frac{4}{10000^{0/512}}), cos(\frac{4}{10000^{0/512}}), sin(\frac{4}{10000^{2/512}}), cos(\frac{4}{10000^{2/512}}), \dots, sin(\frac{4}{10000^{512/512}}), cos(\frac{4}{10000^{512/512}})]$$

6. Self-Attention特点？为什么有效?

- 无视词(token)之间的距离直接计算依赖关系，相比于RNN等，更容易捕获长远距离的相互依赖的特征。
- 不考虑时序关系，可以并行运算，效率高（通过position encoding能够捕获词的时序关系）

7. Decoder阶段的多头自注意力和encoder的多头自注意力有什么区别?

> Decoder有两层多头自注意力层

Decoder的第二层多头自注意力与encoder的输出进行交互，其KV来自Encoder的输出，Q来自Decoder第一层的输出。这样同时能转化输入与输出句长，Q与KV的squence长度不一致，最终得到输出为Q的长度。

8. Transformer相比于RNN有什么优势?

- 无视词(token)之间的距离直接计算依赖关系，相比于RNN等，更容易捕获长远距离的相互依赖的特征。
- 计算时不考虑时序关系，可以并行运算，效率高（并且通过position encoding能够捕获词的时序关系）
- 传统的RNN存在梯度消失和梯度爆炸的问题。LSTM和GRU都缓解了梯度消失和梯度爆炸的问题，但是没有彻底解决该问题
- 当hidden > sequence , transformer 比 RNN 时间复杂度低 详解
  - O(Transformer)： $sequence^2 * hidden$
  - O(RNN)： $sequence * hidden^2$
  - O(CNN): $kernel * sequence * hidden^2$

9. Decoder 可以并行运算吗?

Encoder端可以并行计算，一次性将输入序列全部encoding出来。

Decoder仅在训练阶段可以并行化，但在预测阶段不是一次性把所有单词(token)预测出来的，而是像seq2seq一样一个接着一个预测出来的。

10. 为什么要除以$\sqrt{d_k}$

随着d_k增大，q与 k的点积结果也在增大，会使得softmax梯度**变小**，甚至消失，所以$\frac{q*k}{\sqrt{d_k}}$

11. 为什么Transformer中要用norm？为什么norm又有用?

norm即layer normlization，归一化通过调节均值方差，将结果尽量落在导数为1的区域，能够有效平衡梯度爆炸或者消失问题

12. 在计算attention score的时候如何对padding做mask操作?

对需要mask的位置设为负无穷，再对attention score进行相加

13. Self-Attention运算时，为什么Q/K使用不同的权重矩阵，为何不直接Q=K

> Q = linear(Q) , K = linear(K), V = linear(V)

如果令Q=K，那么得到的模型大概率会得到一个类似单位矩阵的attention矩阵，**这样self-attention就退化成一个point-wise线性映射**。

这样的话，你会发现attention score 矩阵是一个对称矩阵，都投影到了同样一个空间，所以泛化能力很差。

14. 简单讲一下Transformer中的残差结构以及意义

encoder和decoder的self-attention层和ffn层都有残差连接。反向传播的时候不会造成梯度消失

15. 简单描述一下Transformer中的前馈神经网络?

> 也就是讲述Transformer的结构

输入嵌入 - 加上位置编码 - 多个编码器层 (多头自注意力 / 点式前馈神经网络 / add & norm) - 多个解码器层 (多头自注意力/ 多头自注意力交互 / 点式前馈神经网络 / add & norm)

16. Transformer用了什么激活函数？有什么优缺点?

用了Relu激活函数

优点：

- 是非饱和型激活函数，有效避免梯度消失问题
- 第一象限梯度始终为1，收敛速度更快
- 只需要做max运算，效率高

缺点：

- 小于0的神经元会被永久置为0，导致部分神经元坏死

17. Transformer 的分词方式

采用**BPE**（byte pair encoding，双字节编码）方式，BERT中使用的WordPiece encoding可以理解为是其变种，都是属于subword相关的分词方式，能够有效平衡OOV问题。

18. Transformer在哪里进行了参数共享?

- Encoder和Decoder间的Embedding层权重共享；
    - encoder / decoder 都来源于同一个词表（即使翻译任务，两个语言不通，也可放在同一个词表中），**嵌入时都只有对应语言的embedding会被激活**，因此是可以共用一张词表做权重共享的。
- Decoder中Embedding层和FC层权重共享；
    - 实际上，Decoder中的**Embedding层和FC层有点像互为逆过程**。Embedding层可以说是通过onehot去取到对应的embedding向量，FC层可以说是相反的，通过向量去得到它可能是某个词的softmax概率，取概率最大词。通过这样的权重共享可以减少参数的数量，加快收敛。