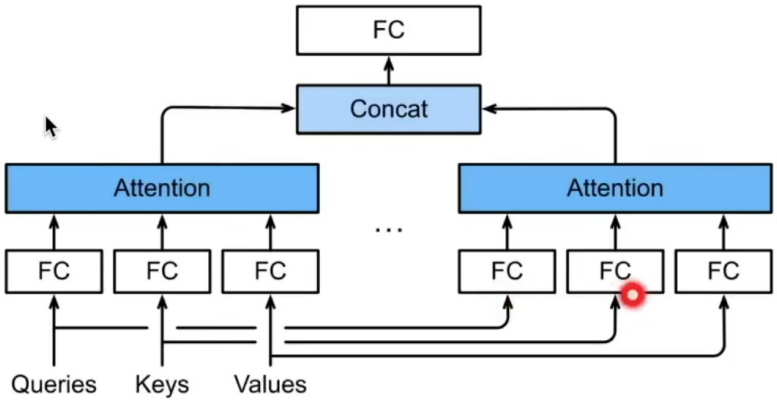
父条目： Attention Is All You Need

Transformer笔记

多头注意力

- 对同一key，value，query，希望抽取不同的信息
  - 例如短距离关系和长距离关系
- 多头注意力使用 h 个独立的注意力池化
  - 合并各个头（head）输出得到最终输出



Masked Self-Attention¶

传统 Seq2Seq 中 Decoder 使用的是 RNN 模型，因此在训练过程中输入因此在训练过程中输入t时刻的词，模型无论如何也看不到未来时刻的词，因为循环神经网络是时间驱动的，只有当t时刻运算结束了，才能看到t+1时刻的词。而 Transformer Decoder 抛弃了 RNN，改为 Self-Attention，由此就产生了一个问题，在训练过程中，整个 ground truth 都暴露在 Decoder 中，这显然是不对的，我们需要对 Decoder 的输入进行一些处理，该处理被称为 Mask。

Mask 非常简单，首先生成一个下三角全 0，上三角全为负无穷的矩阵，然后将其与 Scaled Scores 相加即可，之后再做 softmax，就能将 -inf 变为 0，得到的这个矩阵即为每个字之间的权重。

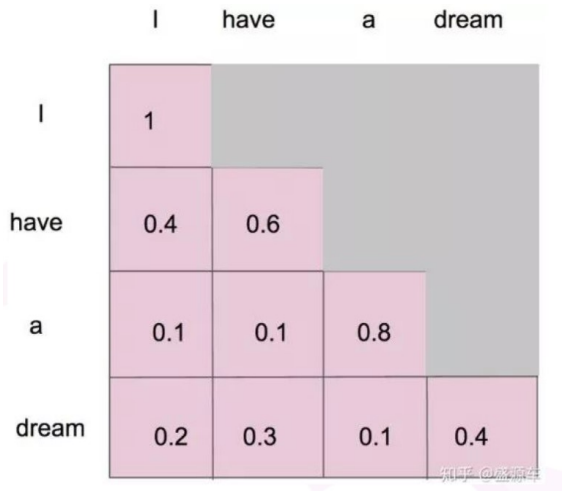
为什么要做这个改进：生成模型，生成单词，一个一个生成的

当我们做生成任务的时候，我们也想对生成的这个单词做注意力计算，但是，生成的句子是一个一个单词生成的

I have a dream

1. I 第一次注意力计算，只有 I
2. I have 第二次，只有 I 和 have
3. I have a
4. I have a dream
5. I have a dream <eos>

掩码自注意力机制应运而生



Position Encoding¶

Transformer模型中还缺少一种解释输入序列中单词顺序的方法。为了处理这个问题，transformer给encoder层和decoder层的输入添加了一个额外的向量Positional Encoding，维度和embedding的维度一样，这个向量采用了一种很独特的方法来让模型学习到这个值，这个向量能决定当前词的位置，或者说在一个句子中不同的词之间的距离。这个位置向量的具体计算方法有很多种，论文中的计算方法如下

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

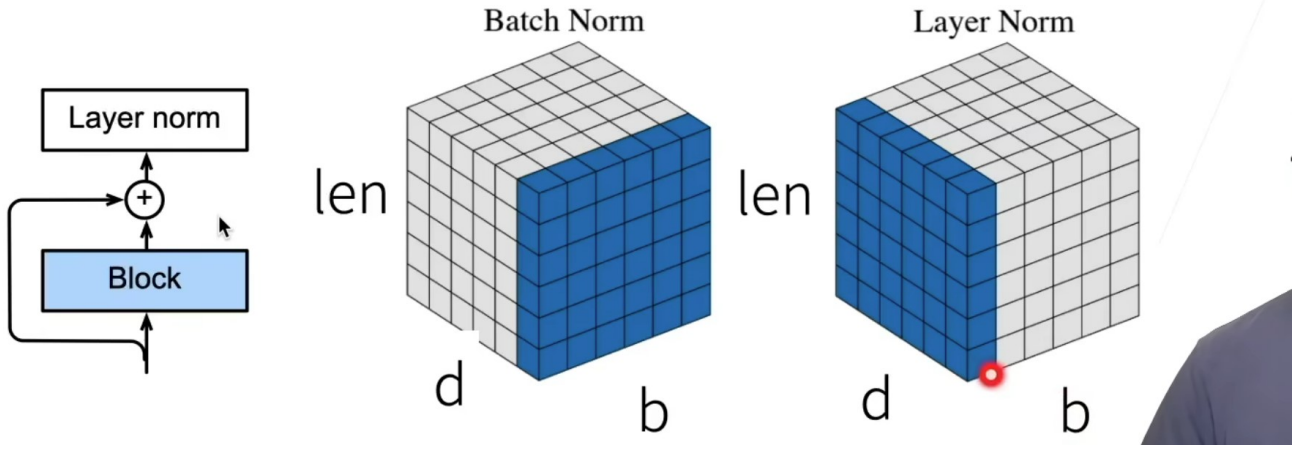
其中pos是指当前词在句子中的位置，i是指向量中每个值的index，可以看出，在偶数位置，使用正弦编码，在奇数位置，使用余弦编码.

层归一化

d:特征向量；b:batch；len: 序列长度

层归一化

- 批量归一化对每个特征/通道里元素进行归一化
  - 不适合序列长度会变的NLP应用
- 层归一化对每个样本里的元素进行归一化

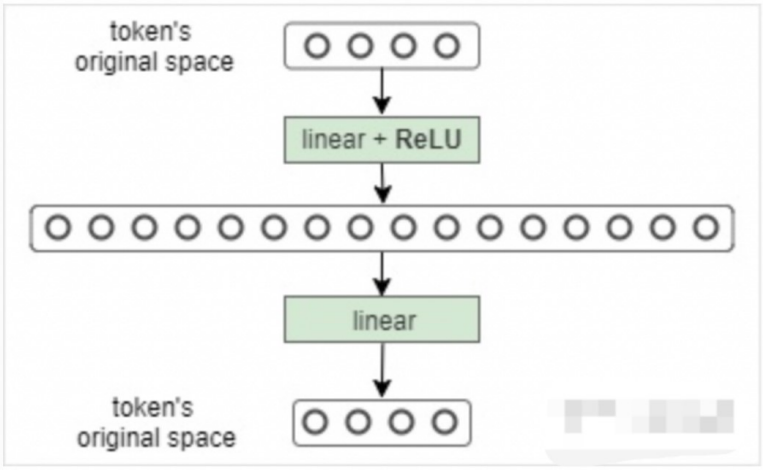


对样本归一化而不是特征. 因为样本是不固定的。

因为每句话的长度不一样即每个b对应的len是会变的，不是一个完整的魔方，而是凹凸不平的。

有b句话，每句话有len个词，每个词由d个特征表示，BN是对所有句子所有词的某一特征做归一化，LN是对某一句话的所有词所有特征做归一化

Feed Forward (Position wise Feed Forward)



将Multi-Head Attention得到的向量再投影到一个更大的空间（论文里将空间放大了4倍）在那个大空间里可以更方便地提取需要的信息（使用Relu激活函数），最后再投影回token向量原来的空间。

借鉴SVM来理解：SVM对于比较复杂的问题通过将特征其投影到更高维的空间使得问题简单到一个超平面就能解决。这里token向量里的信息通过Feed Forward Layer被投影到更高维的空间，在高维空间里向量的各类信息彼此之间更容易区别。

注意区别: 原始的MLP处理的是二维数据例如（128，512）,而这个PositionwiseFeedForward 中的Position体现在此处输入的不是二维的数据而是三维的例如(128, 30, 512), 30就是序列的长度。其本质依然是对512的特征维度进行MLP但是输入数据不同。

预测(推理)

预测

- 预测第 $t+1$ 个输出时
- 解码器中输入前 t 个预测值
  - 在自注意力中，前 t 个预测值作为key和value，第 t 个预测值还作为query