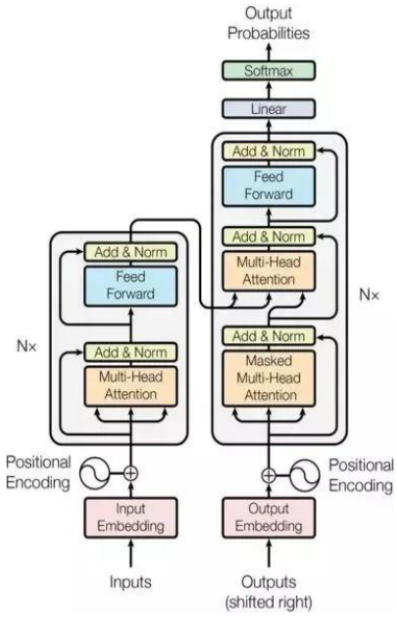
第一章

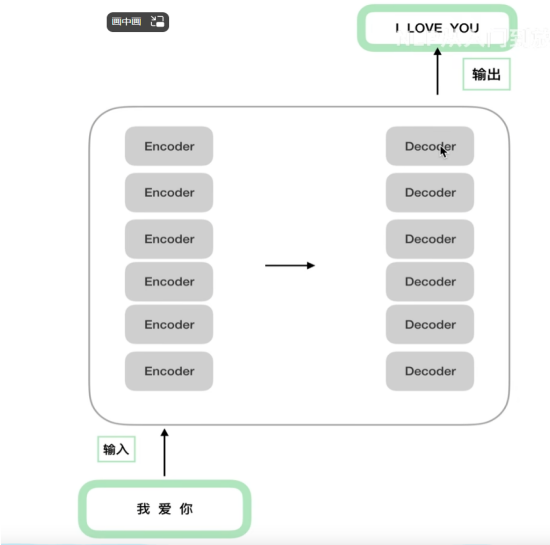
1.从全局概括Transformer



左测为encoder 右侧为decoder

Transformer是由谷歌于2017年提出的具有里程碑意义的模型，同时也是语言AI革命的关键技术。在此之前的SOTA模型都是以循环神经网络为基础（RNN, LSTM等）。从本质上来讲，RNN是以**串行的方式**来处理数据，对应到NLP任务上，即按照句中词语的先后顺序，每一个时间步处理一个词语。

相较于这种串行模式，**Transformer的巨大创新便在于并行化的语言处理**：文本中的所有词语都**可以在同一时间进行分析**，而不是按照序列先后顺序。为了支持这种并行化的处理方式，**Transformer依赖于注意力机制**。注意力机制可以让模型考虑任意两个词语之间的相互关系，且不受它们在文本序列中位置的影响。**通过分析词语之间的两两相互关系，来决定应该对哪些词或短语赋予更多的注意力。**



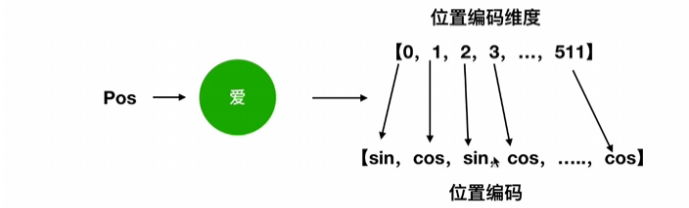
- *encoder*和*decoder*的结构不同，各*encoder*之间的结构相同但参数不同，同理，各*decoder*之间的结构相同但参数不同。
- 二者的数量可以通过两个超参数来调整
- 在后期有些预训练模型中将*encoder*中某些层的参数进行共享，对*decoder*同理，进而达到减少参数量的目的

2.位置编码的作用

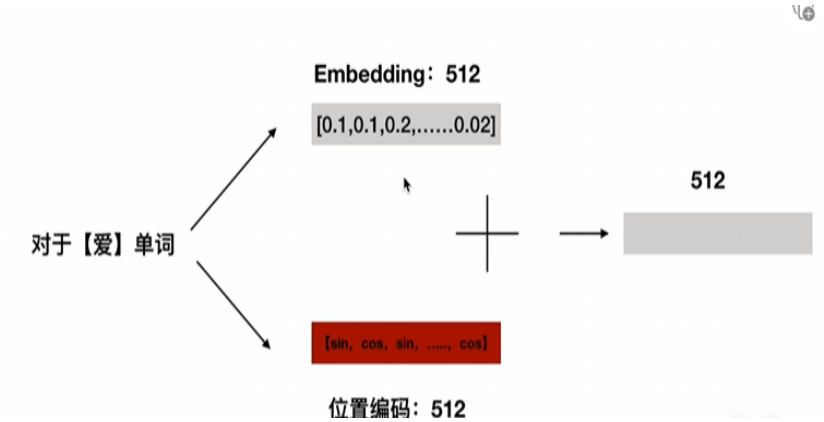
- 对于**RNN**的不同*time steps*，都共享一套参数，即输入参数、输出参数、隐层参数等
- **RNN**与普通网络的**梯度消失**并不同，它的梯度消失是一个总和，总梯度被近距离梯度主导，而由于不同时间步之间参数共享，进而远距离的梯度往往出现在一长串的连乘之中，故远距离梯度可能被大幅度减小或者方法，进而引出**RNN**的梯度消失和梯度爆炸问题
- ***batch normalization***一般无法运用在**RNN**中，因为归一化是“垂直”应用的，即每个时间步上**RNN**的输出。但是**RNN**是水平的，例如在时间步之间，他会因为重复的重新缩放而产生爆炸性的梯度，进而伤害到训练
- ***transformer***与**RNN**不同的地方在于，它是同时处理不同时间步的信息的，可以理解为对N天的股票序列同时处理，但这样肯定会相比**RNN**缺少对序列依赖关系的感知（因为**RNN**是一个一个处理的，可以学习到这种天然的序列或者说前后关系，如NLP中单复数用was/were,股票序列中会不会因为前一天的股价而对今天的股价产生影响等），所以我们需要**位置编码**
- **位置编码**的作用是：
  当我们将一段文本分词并转化为词向量数组时，词的顺序在自然语言中确实扮演着重要的角色。**这是因为在不同的语境下，词语的排列顺序可以传达不同的意义和语义。**这个问题在自然语言处理中被称为“**序列建模**”问题.为了更好地捕捉这种语义信息，我们引入位置编码器。**位置编码器的作用是为词向量添加一些位置相关的信息，以便模型能够区分不同位置的词语。它通过将位置信息与词向量相结合，从而使词向量变得更加丰富和具有上下文感知能力。**
  位置编码器通常**使用正弦和余弦函数等数学函数来生成位置编码。这些编码会根据词在句子中的位置而变化，因此不同位置的词会具有不同的位置编码。**通过将位置编码添加到词向量(**即与embedding相加**)中，模型可以区分不同位置的词，从而更好地理解文本的语义。（**故：位置编码维度应与embedding维度相同**）

位置编码公式

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



其中pos为位置编码中的不同位置



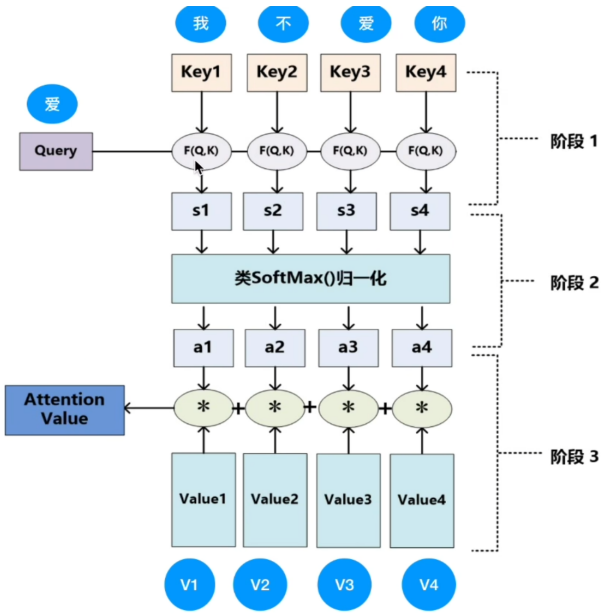
位置编码：512
将经由sin/cos计算所得位置编码与embedding相加，得到最终的输入embedding

3.多头注意力机制

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

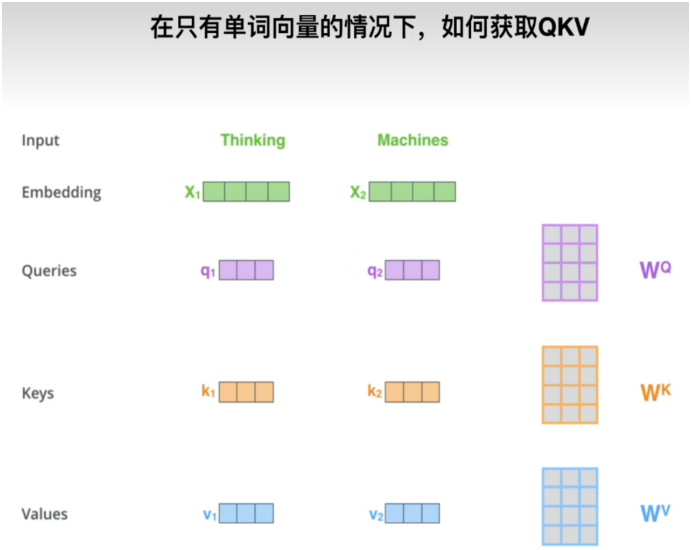
注意力机制公式

- 注意力机制必须要有**Q K V**三个向量或矩阵



注意力机制结构

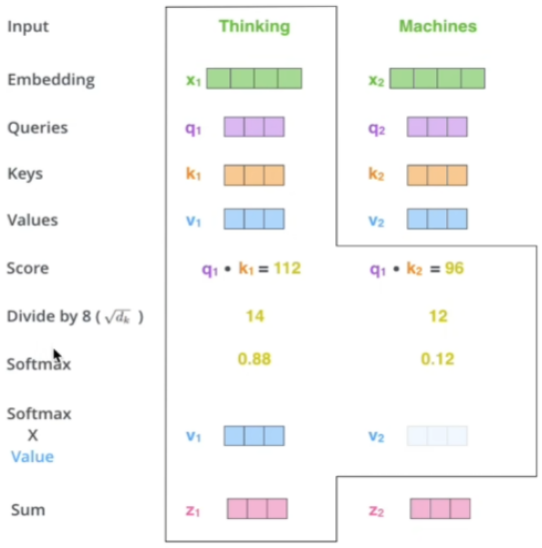
- **F（Q，K)代表两向量点乘或矩阵相乘，向量点乘代表两个向量逐个元素相乘再相加，也可以代表一个向量在另一个向量上的投影，两个向量越相似，则两个向量的点乘就应该越大**，这也是毕设中GCN的设计思路
- 通过softmax的激活值a1-4，再分别与V1-4相乘后再相加，即softmax所得出的向量与向量V点乘，得出最后的***Attention Value***



如何获取KQV三个向量/矩阵

- 例如在NLP中，Thinking对应的**embedding**为 X_1,将其分别与 W_Q、W_K、W_V 相乘然后获得**Q K V**三个向量,再将其带入到缩放点积公式中进行计算
- 注意：**如果Q = K = V**，即上述W_Q、W_K、W_V 相同，那么这就是自注意力机制***self attention***

计算QK相似度，
得到attention值



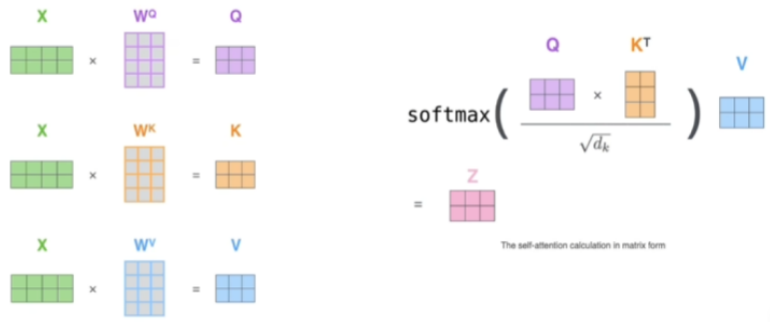
特征词的向量	计算公式1	计算公式2
Queries	q1 = x1 · WQ	q2 = x2 · WQ
Keys	k1 = x1 · WK	k2 = x2 · WK
Values	v1 = x1 · WV	v2 = x2 · WV
Score	s1 = q1 · k1=112	s2 = q2 · k2 = 96
Divide by 8	d1 = s1 / 8 = 14	d2 = s2 / 8 = 12
Softmax	sm1 = e14/ (e14 + e12) = 0.88	sm2 = e12 / (e14 + e12) = 0.12
Softmax * value	v1 = sm1 * v1	v2 = sm2 * v2

计算过程实例

- 为何除以$\sqrt{dk}$:
  1.具体来说，在计算注意力得分时，通常会使用一个**相似性度量函数来衡量输入序列中不同位置之间的相似程度(即Q✕K^T)**。这个相似性度量函数的**输出值**通常会受到**输入序列的长度和特征维度的影响。**
  2.通过除以$\sqrt{dk}$力得分可以**将注意力得分的尺度归一化到一个相对稳定的范围内**，使得不同位置和维度的注意力得分具有可比性。这样可以确保在进行后续的softmax 操作或其他与注意力得分相关的计算时，不同位置的注意力得分**不会因为输入序列的长度或特征维度的变化而产生过大的差异,且不会因为Q*K^T数值过大而落入softmax的小梯度区间进而造成梯度消失。**
  3.此外，除以$\sqrt{dk}$还可以在一定程度上**减轻由于输入序列长度和特征维度的增加而导致的数值不稳定性问题。**这样可以提高模型的训练稳定性和泛化能力。

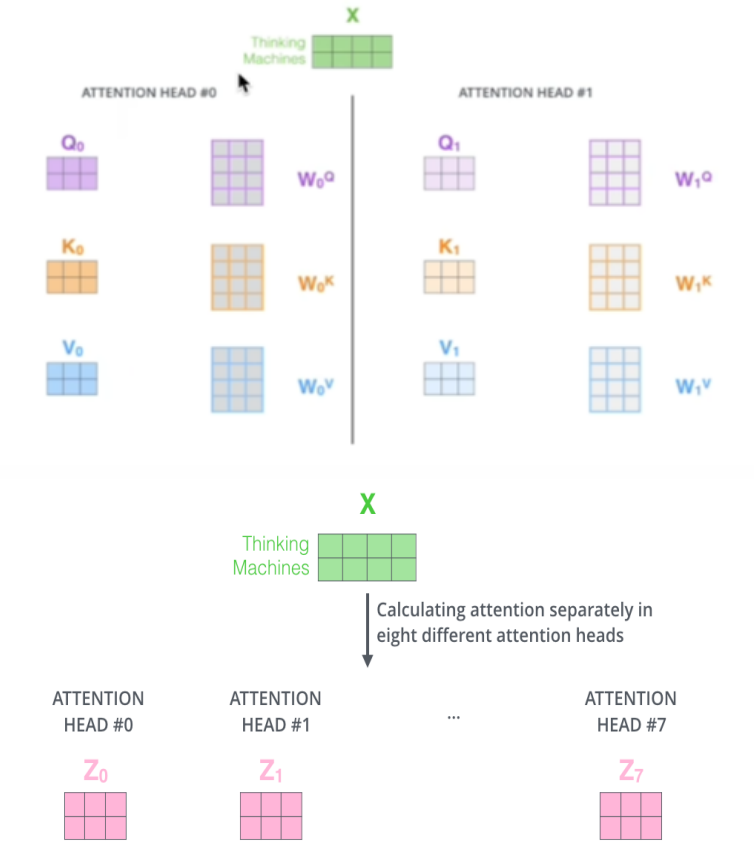
- 为何除以$\sqrt{dk}$而不是*dk*: **原始表征是符合均值N(0,1)的，而与权重矩阵相乘后，结果符合N(0,dk)的正态分布了，所以为了不改变原始表征的分布，需要除以$\sqrt{dk}$。这可以更好地控制数值的量级,使得注意力得分在不同的位置和维度上具有相对统一的分布，恒为N（0，1），有助于避免数值溢出或下溢的问题**，提高计算的稳定性。

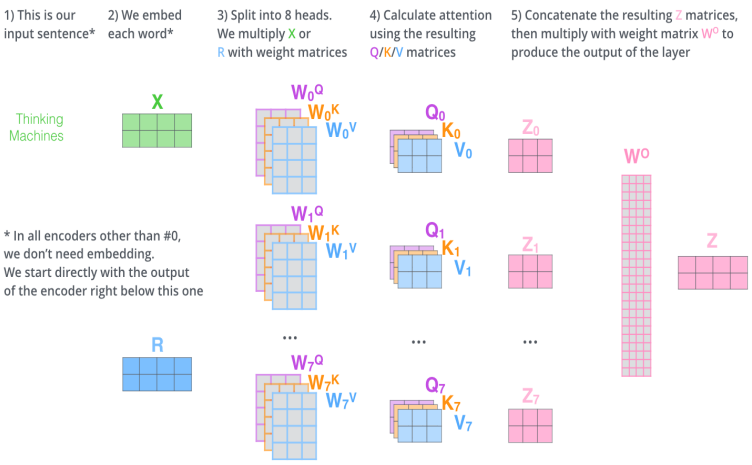
实际代码使用矩阵，方便并行



实际代码使用矩阵

- 图中X可以理解为**上图thinking和machines所对应的embedding拼接到一起,通过X与三个不同的针对性权重矩阵相乘得出Q K V三个矩阵**
- **三个矩阵的维度可以用权重矩阵来控制,三个矩阵的值可以随机初始化,而dimension是自己规定的**
- 1.由于X为所有embedding拼接到一起，**所以才所说它是同时处理不同时间步的信息的：它可以同时看到不同的embedding，而不是像RNN或者LSTM那样一次只能看到一个或者n个embedding,要经历多个时间步才能看完。**
  2.可以理解为：**毕设中，单只股票每天embedding的deminsion为[1✕5]，将N天的emdedding拼接到一起形成一个大的X，deminsion为[N✕5]，再输入到tranformer中，对N天的股票序列同时处理.**
  3.这有助于理解下文Decoder中的***Mask Muti-head Attention***
- Q✕K^T代表了**相似性度量函数,可以得出相似度得分**,进而归一化后通过softmax与**V**相乘得出注意力得分**Attention value**
- 得出**Attention value**的过程被叫做**缩放点积模型**，具体请参考(试图带你一文搞懂transformer注意力机制（Self-Attention）的本质_transformer中attention(q,k,v)的本质事什么-CSDN博客)（网址：https://blog.csdn.net/athrunsunny/article/details/133780978）





多头注意力机制

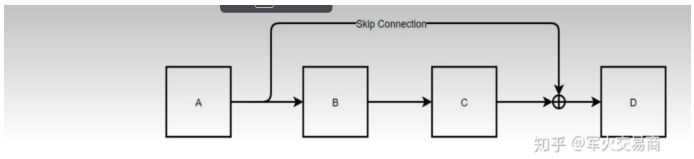
- **多头注意力机制**:
  1.使用**多套计算Q K V的针对性权重矩阵,通过此种方法计算出不同的Z**.
  2.**Concatenate所有的Z_i矩阵，然后乘以一个权重矩阵W^O**（随机初始化所得）,最终计算所得一个矩阵Z，作为多头注意力部分的输出

- 使用多头注意力机制具有**以下几点好处**:

1. **多角度信息融合**：***多头注意力机制通过多个头（多个独立的注意力子机制）来处理输入序列，每个头可以关注输入序列的不同子空间或特征表示。这样可以捕捉到输入序列中的多种信息，从而提供更全面和丰富的表示。(more important)***
2. **增强模型的灵活性和适应性**：不同的头可以学习到不同的注意力模式和特征，从而增强了模型对不同任务和数据的适应性。模型可以根据输入的特征和任务需求，灵活地调整不同头的权重和关注重点。
3. **提高模型的容量和表达能力**：多个头的设计增加了模型的参数量和计算量，从而提高了模型的容量和表达能力。这使得模型能够处理更复杂的任务和学习更复杂的模式。
4. **并行处理和效率提升**：***多头注意力机制可以在不同的头之间进行并行计算，提高了计算效率。这对于处理大规模数据和需要实时响应的任务非常重要。(more important)***
5. **捕获长距离依赖关系**：通过多个头的协同作用，**多头注意力机制能够更好地捕捉输入序列中的长距离依赖关系。**这对于处理自然语言等具有长序列结构的任务非常有益。
6. **增强模型的鲁棒性**：多个头的设计可以在一定程度上减轻过拟合的风险，因为不同的头可以学习到不同的模式和特征，从而增加了模型的鲁棒性。

- **注意力机制的缺点**：没法捕捉位置信息，即没法学习序列中的顺序关系。这点可以通过加入位置信息，如通过位置向量来改善，具体如bert模型。

4.残差网络Resnet为何可以减弱梯度消失



A,B,C,D 为四个不同的网络块，箭头代表"数据流"

根据后向传播的链式法则，

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} \frac{\partial X_{Din}}{\partial X_{Aout}}$$

而 $X_{Din} = X_{Aout} + C(B(X_{Aout}))$

所以:

$$\frac{\partial L}{\partial X_{Aout}} = \frac{\partial L}{\partial X_{Din}} [1 + \frac{\partial X_{Din}}{\partial X_C} \frac{\partial X_C}{\partial X_B} \frac{\partial X_B}{\partial X_{Aout}}]$$

残差网络与梯度反向传播链式法则公式

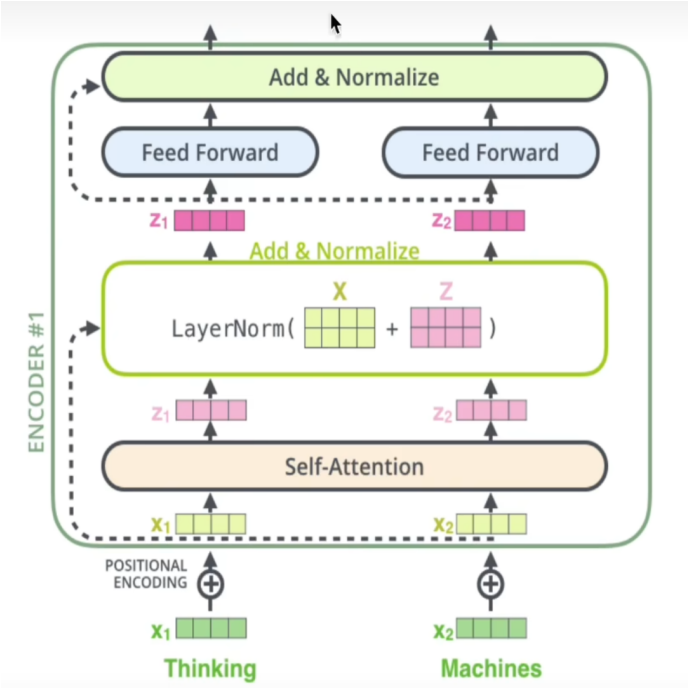
- 可以看到，这是一个残差块且长度为2，A的输出要与C的输出相加，再作为输入数据进入D。根据链式求导法则，可与推导出损失函数L对 X_{Aout} 的求导公式，我们知道，**梯度消失往往因为许多小于1的梯度进行连乘，进而导致结果x无限接近于0，模型无法有效更新参数。而残差网络Resnet将**

两步之前的输出与当前的输出相加再输入到下一步长中，所以在求导公式中会出现1+x的这种形式，这样保证了其不会等于0，而是在1的邻域内，所以可以有效避免梯度消失.

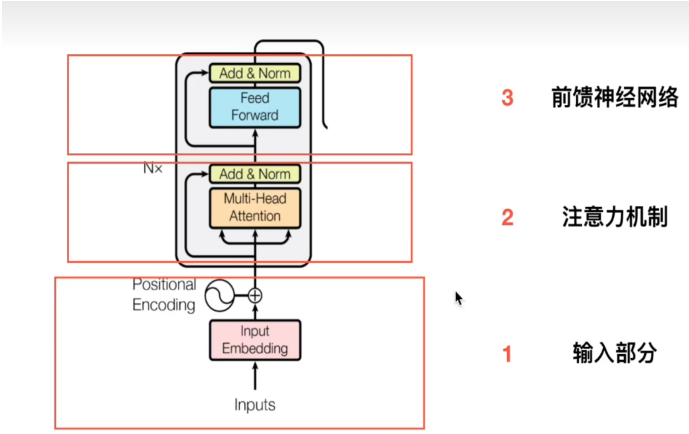
5.为什么在transformer中不使用Batch normalization而使用layer normalization

- **BN的优点**:
  1.**解决内部协变量偏移**：各隐藏层的输出作为下一层的输入，**而不同隐藏层的输出具有不同的分布**，这样就增大了学习难度。使用BN可以**人为的使用两个超参数α、β来让不同隐藏层的输出在同一分布上（正态分布，先将输出值标准正态化为N（0，1），再让其乘以α后加上β，这样原来的数据就被正态化为N（β，α^2））,如此保证了前层不会左右移动的太多**.
  2.缓解梯度饱和问题：若超参数α，β选择适当，可以有效加快收敛，加快模型训练速度（参考feature scaling）。
- BN的缺点:
  1.由于**BN是基于mini_batch的，当batch_size比较小的时候，我们需要用一个较小的局部样本的均值和方差来模拟整个样本的均值和方差**，来训练模型和更新参数，所以可能让模型学习到噪音，进而减缓模型收敛速度。
  2.BN一般无法运用在**RNN**中，因为归一化是“垂直”应用的，即每个时间步上**RNN**的输出。但是**RNN是水平的，输入是动态的**，它会因为重复的重新缩放而产生爆炸性的梯度，进而伤害到训练（可以理解为毕设中对每支股票再某一时间步上输出的激活值进行归一化后再输入到下一时间步上与下一时间步的输入进行计算）
- 使用LN而不是BN的原因：输入序列的长度问题。在NLP中，我们无法要求每一个序列的长度相同；在CV中，虽然特征会经过padding处理而达到同意的维度，但是padding的0值其实是无用信息，实际上有用的信息还是序列信息，而不同序列的长度不同，所以这里不能使用bn一概而论

6.Encoder结构详解

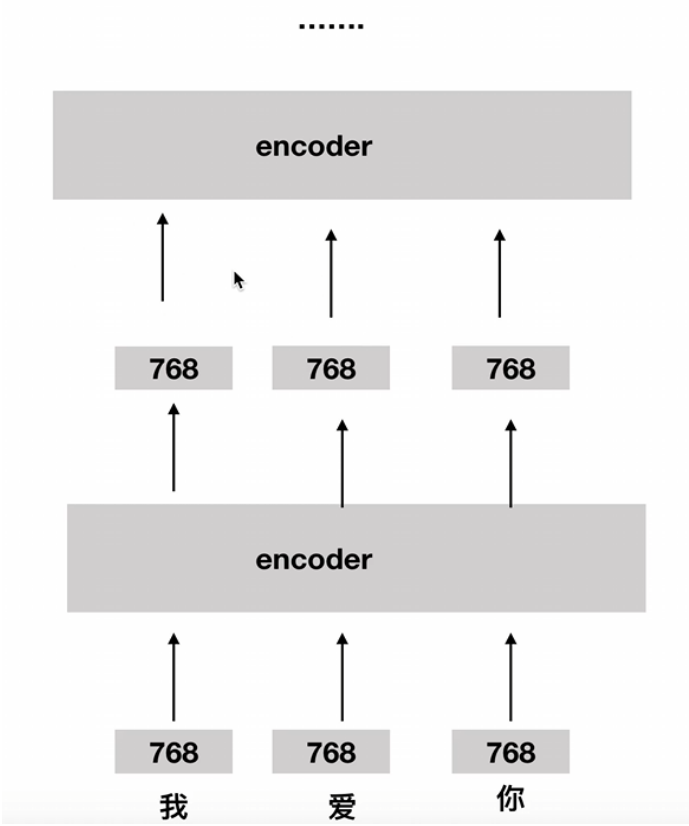


Encoder三个组成部分示意图1



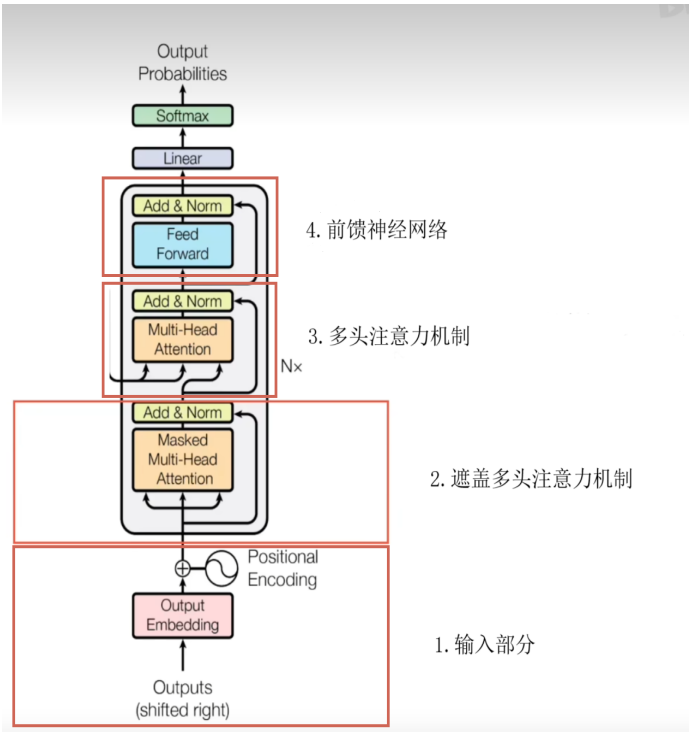
Encoder三个组成部分示意图2

- **输入部分**：将输入的token embedding，记为X_0，与上文方法所得位置编码positional embedding(**偶数位置用正弦，技术位置用余弦**）相加，得出新的输入embedding作为下一步输入，记为X_1
- **多头注意力机制部分**：先将X_1输入到缩放点积注意力模型中，且应用多头注意力***muti-head attention***，通过多套且三个不同的针对性权重矩阵计算出Q_i，K_i，V_i三个矩阵(i代表套数，即muti-head中的头数)，进而计算出不同的Z_i. 然后使用resnet残差网络的思想，将 X_1 与所得各Z_i相加，且通过层归一化layer normalization(LN),得出注意力机制部分的输出，将此操作记为***res+LN***,所得输出记为Z'_i
- **前馈神经网络部分**：
  1.简单的将上一部分所得Z'_i输入全连接层进行向前传播，再通过一个***res+LN***操作，得到此Encoder的输出,记为Z''_i
  2.全连接层公式：$\mathbf{w_2 \times [relu(w_1 \times x+b_1)]+b_2}$,这里使用FFN层的原因是：为了使用非线性函数来拟合数据。如果说只是为了非线性拟合的话，其实只用到第一层就可以了，**但是这里为什么要用两层全连接呢? 使用第二层全连接层是为了进行维度变换**
- **Encoder不改变数据的格式，输入和输出的格式相同**，下面为具体例子：



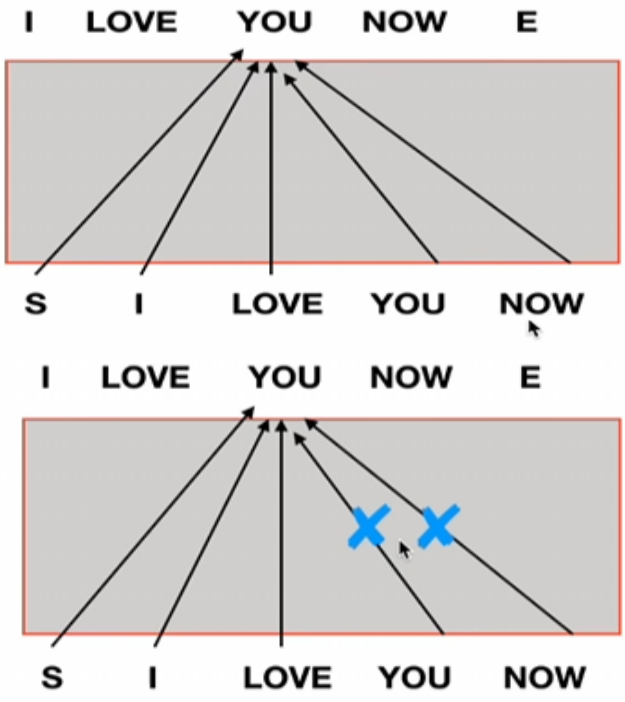
Encoder不改变数据的格式

7.Decoder结构详解



Decoder四个组成部分示意图

- **遮盖注意力机制**：它的作用就是**防止在训练的时候使用未来的输出的单词。**比如训练时，**第一个单词是不能参考第二个单词的生成结果的，此时就会将第二个单词及其之后的单词都mask掉。**总体来讲，mask的作用就是用来**保证预测位置i的信息只能基于比i小的输出。**因此，**encoder层可以并行计算，一次全部encoding出来，但是decoder层却一定要像RNN一样一个一个解出来，因为要用上一个位置的输入当做attention的Q矩阵**.

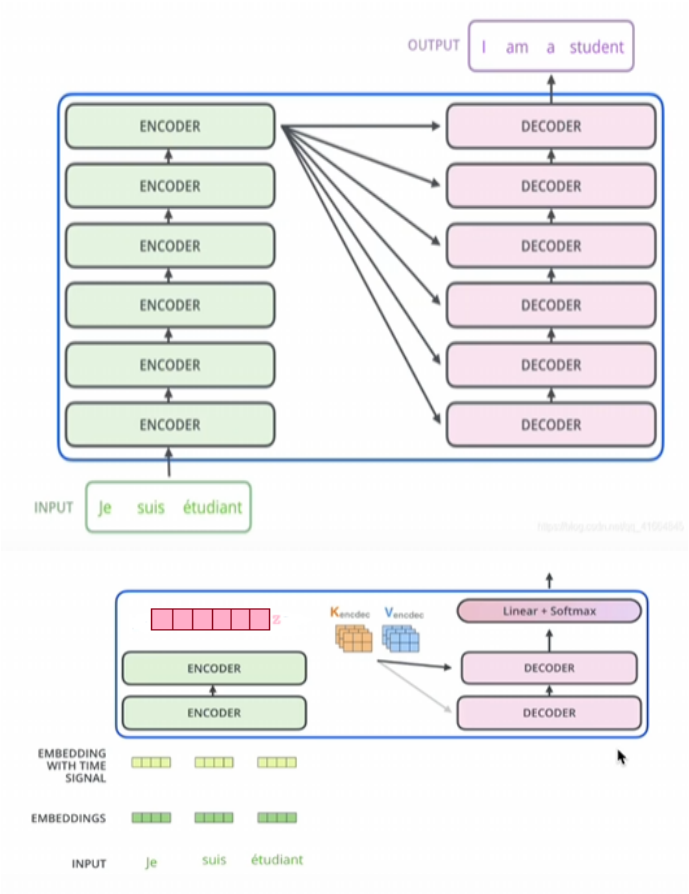


遮盖机制简单举例

- 假设在NLP的文本预测领域，矩阵上方为句子预测结果，下方为训练时Decoder的有序输入，当我们输入到LOVE的时候，模型应当输出YOU。
- 若不使用遮盖机制，则形如图片上半，模型在训练中，**基于muti-attention并行输入机制，YOU的输出预测是会受到输入中LOVE后面YOU和NOW影响，二者都会给YOU的预测提供信息。而在验证和测试过程中，模型是无法并行知道全部的输入的，即：训练看得到后续，而验证预测看不到后续**
- 使用遮盖机制之后，模型在训练时也看不到后续的输出。形如图片下半部分，我们不让输入中的YOU和LOVE为YOU的输出预测提供信息。如此操作，我们就很好的消除了模型在训练和验证测试阶段的gap。

- Decoder最后使用一个FC（Full connection,即全连接）的linear层进行输出，不使用激活函数，但却不使用单纯的dense层（FC层）进行输出的原因，在于损失方式CrossEntropy：该损失函数集成了log_softmax和nll_loss。因此，相当于FC层后接上CrossEntropy，实际上是有经过softmax处理的。只是内置到损失函数CrossEntropy中去了。

8.Encoder和Decoder的消息传递机制



Encoder与Decoder的消息传递

- 需要注意的是：最后一个Encoder的输出为Z(由Concatenate多头注意力输出的Z_i后乘以权重矩阵W^O，再通过前馈神经网络的两层全连接层后输出到下一Encoder，如此循环N次得出最后一个Encoder的输出Z)，尤其计算出的K和V矩阵，输入到每个Decoder中作为多头注意力机制中的K，V矩阵，而Decoder中的Q矩阵则是由自我的输入得出，即**Decoder的Muti-attention中，K，V由最后一个Encoder的输出Z计算所得，Q来自自身。**

9.完整模型细节展示