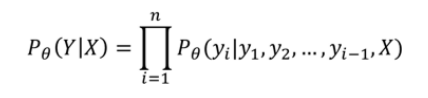
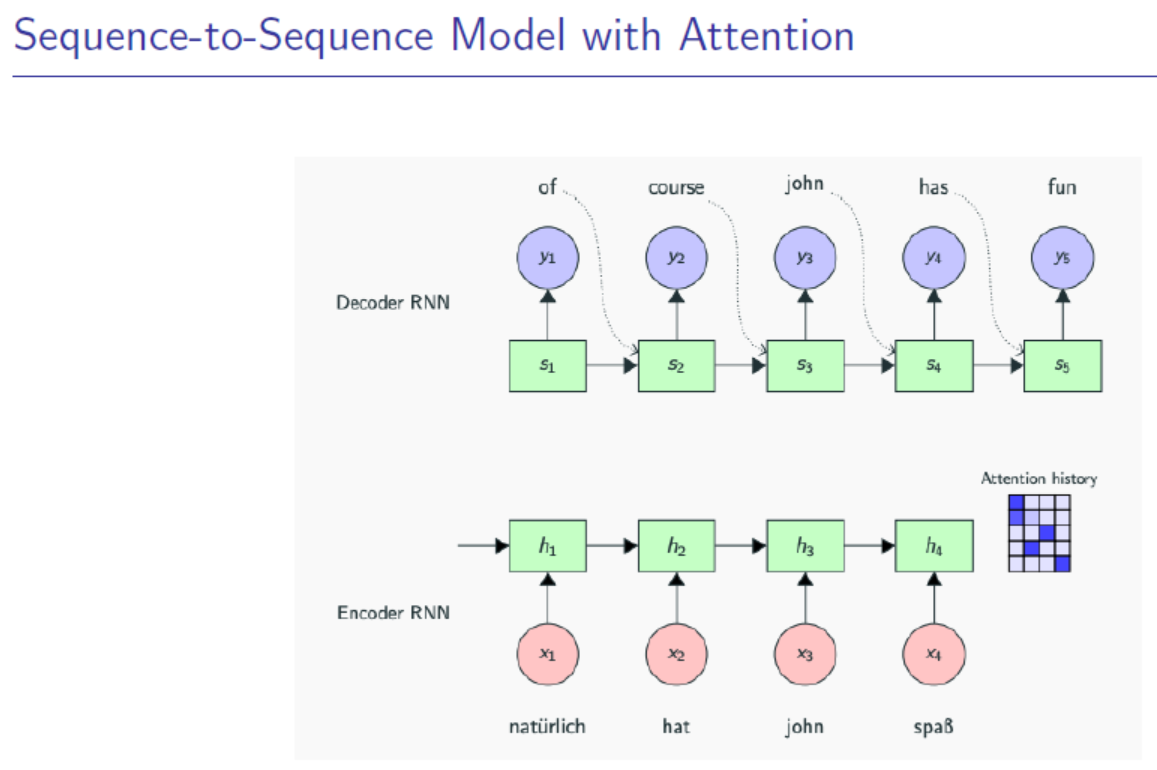
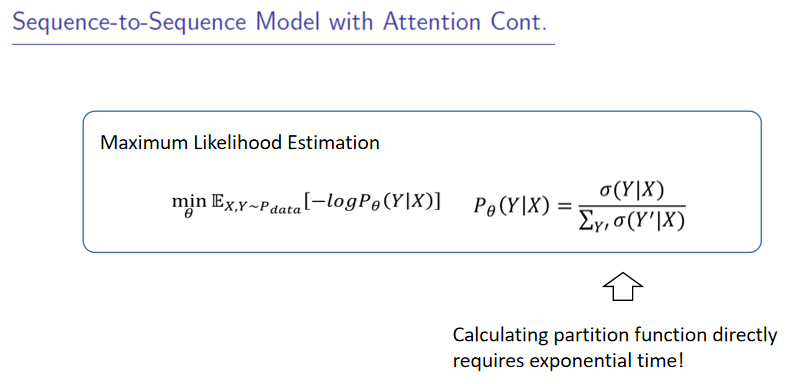
## Transformer model

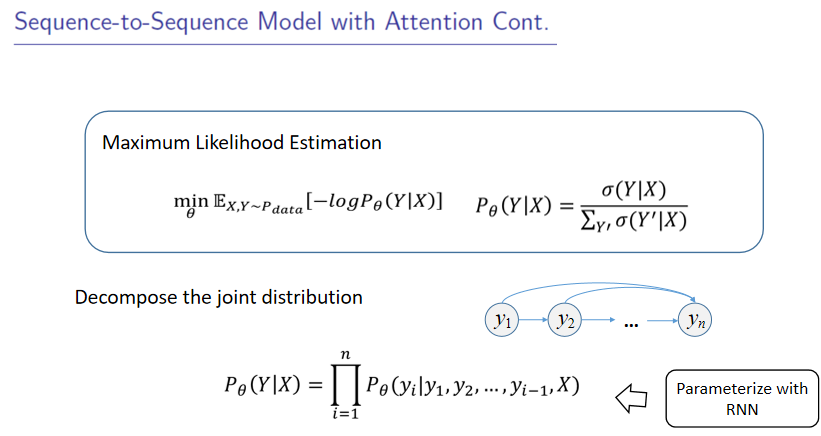


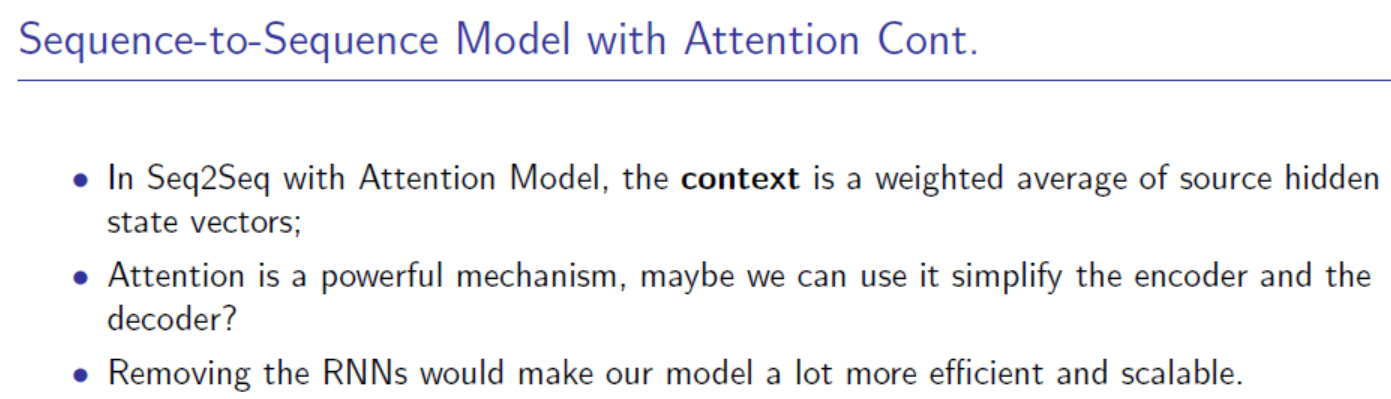
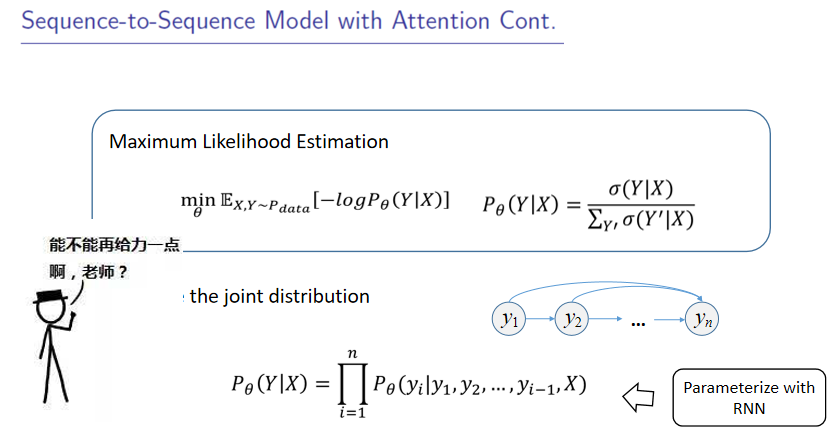
用decoder的输入做Query（y\_i），用encoder的输出去做key和value(X)

## 引入







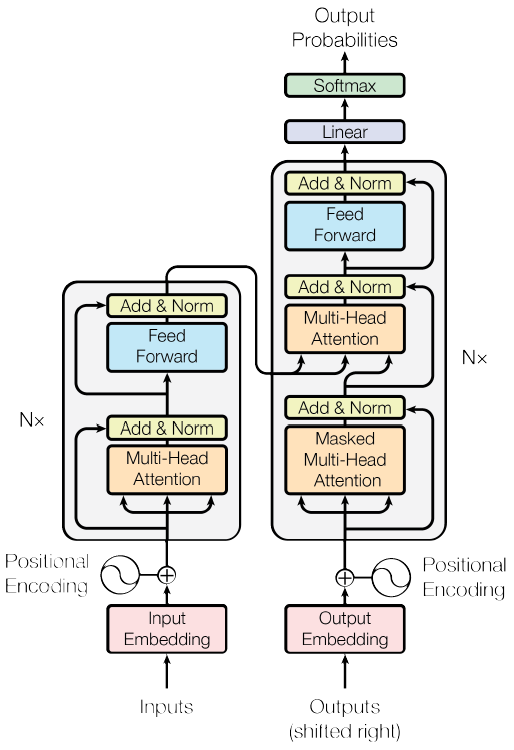


抛弃RNN，只用attention

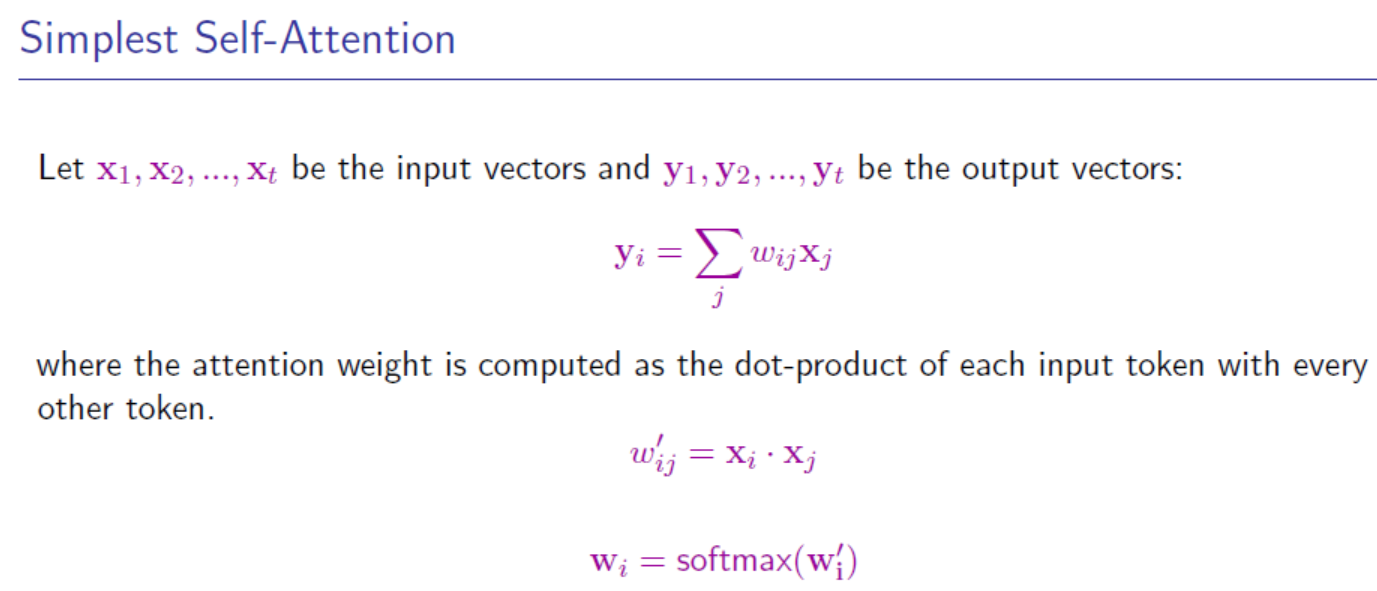
## Transformer ：Attention is All You Need

你不需要RNN，你只需要attention就可以参数化语言模型

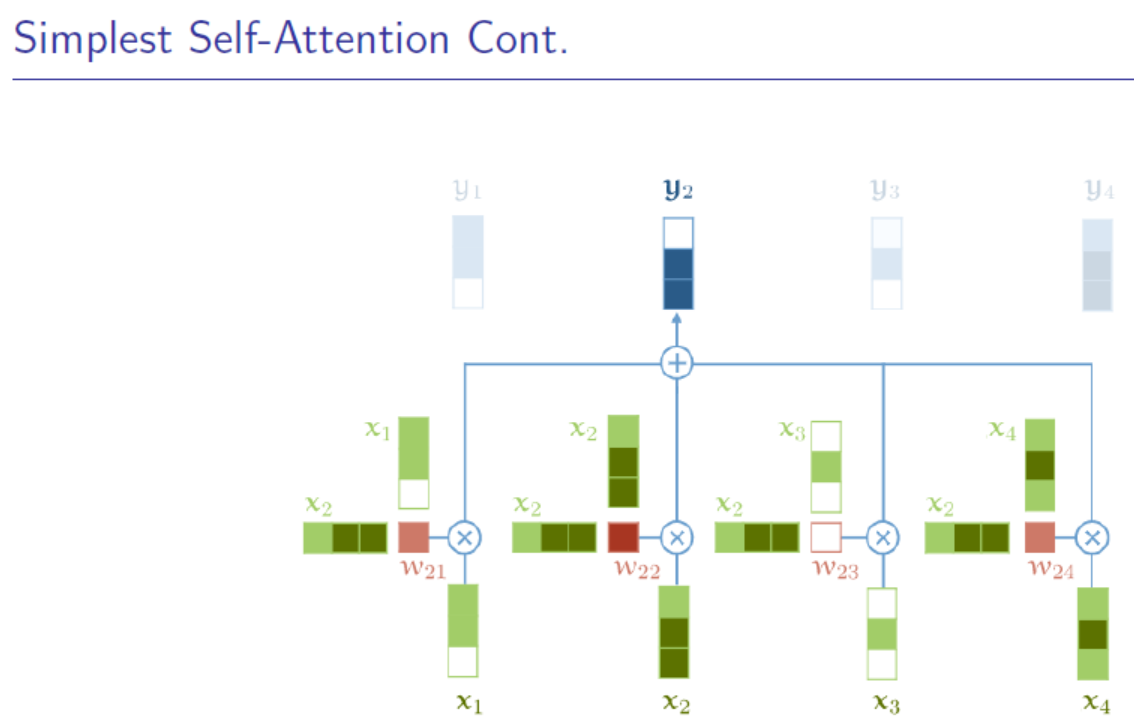
还是有encoder+decoder（基于self-Attention）



## Self-Attention

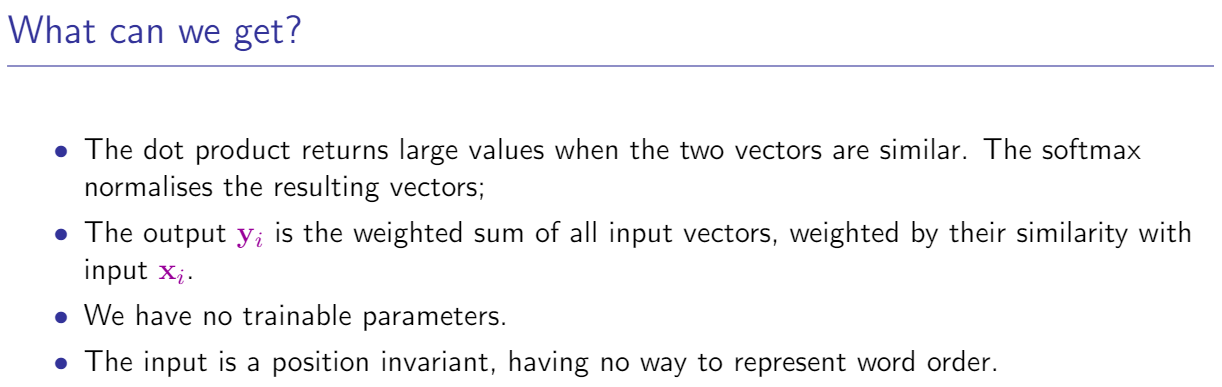


Softmax、加权平均（见上图）



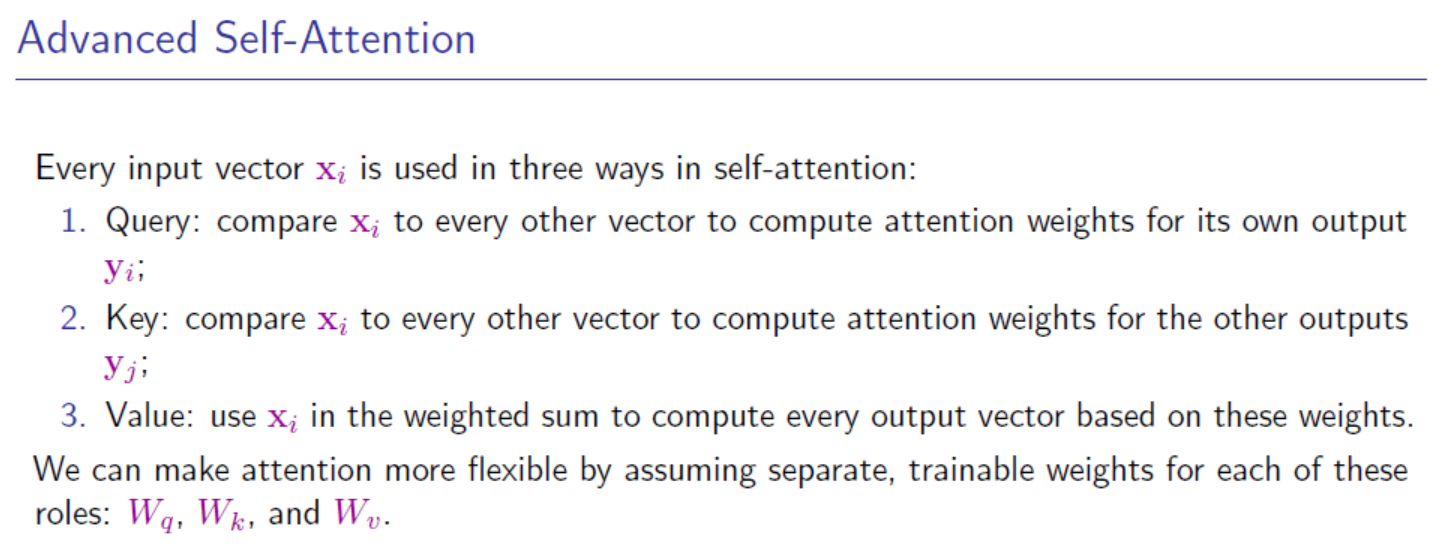
Self-attention得到什么，能有什么好处：**不用额外引入变量**

在nlp中的缺点：x的位置改变的时候，会带来顺序问题（后续在引入self-attention的到transformer的时候，会引入位置编码）**（可以尝试拼接加上自己x的向量，但是输出的y会变长，在cuda中计算也会更加困难）**

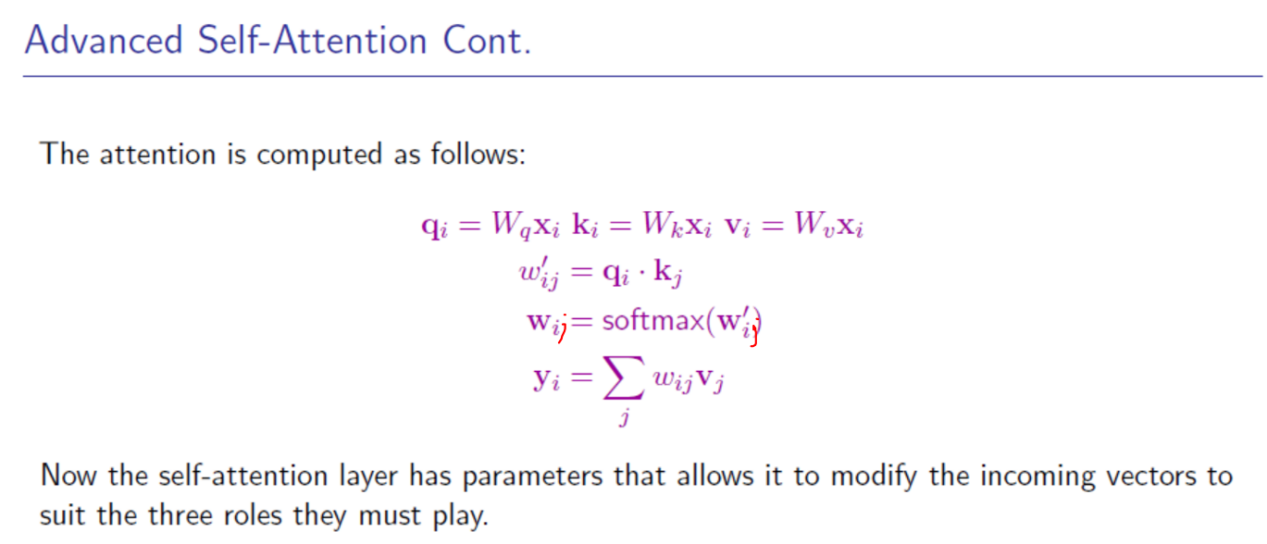


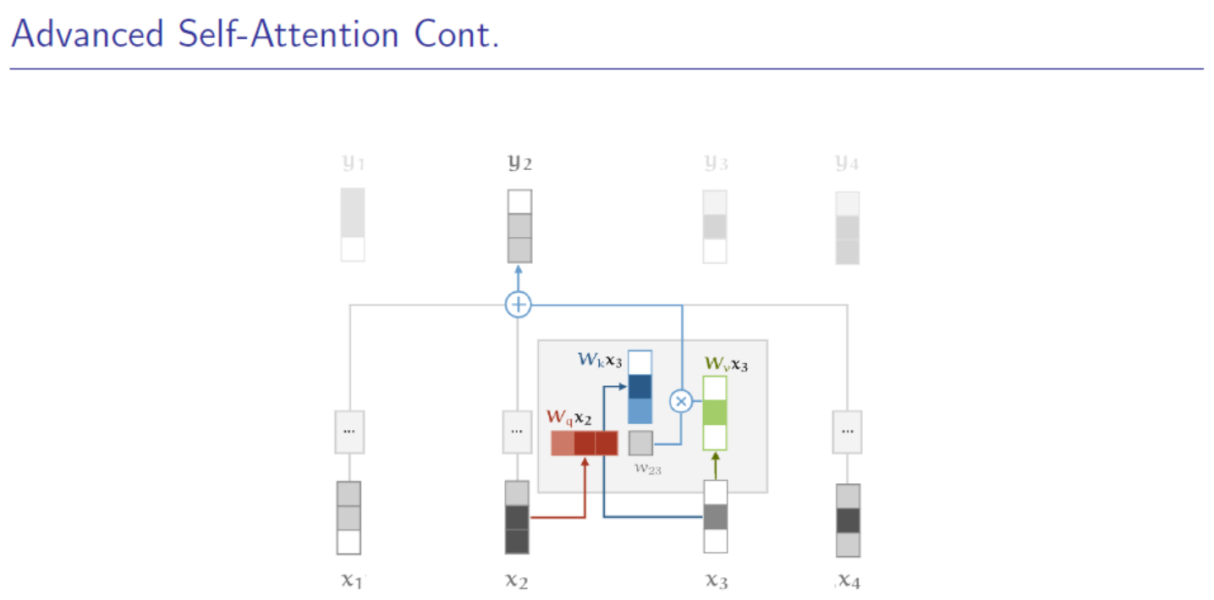
### 尝试区分Query Key Value

在不同的模型中query、key、value可能相同可能不同



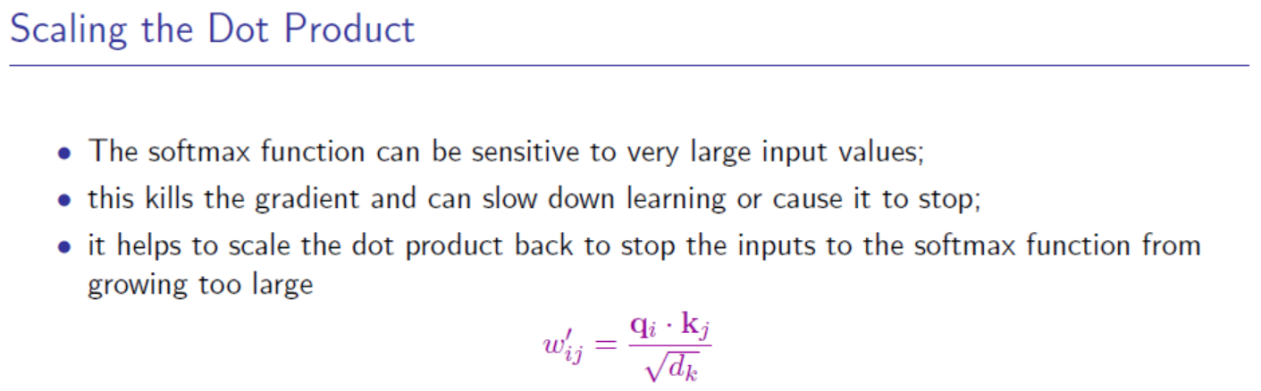
先做线性变换，再做self-attention



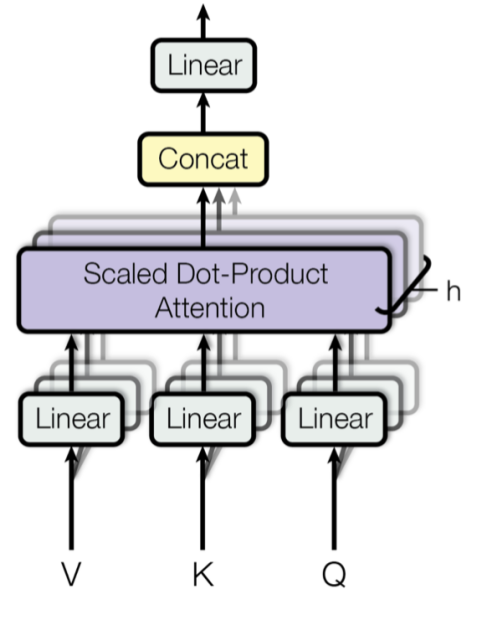


### transformer 中的trick：Scaling the Dot Attention

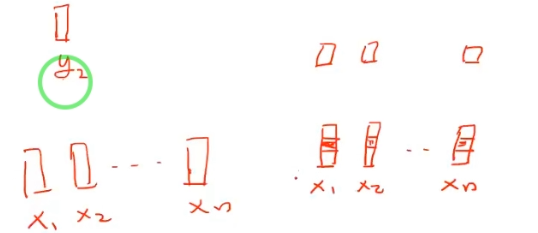
求q、k、v的w\_{ij}的时候用的softmax，softmax很大就会很敏感（梯度消失），解决办法：

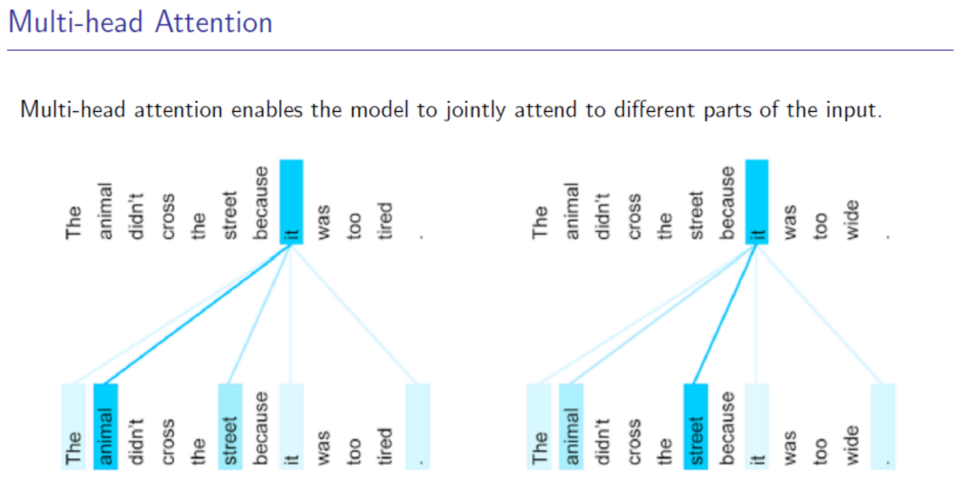


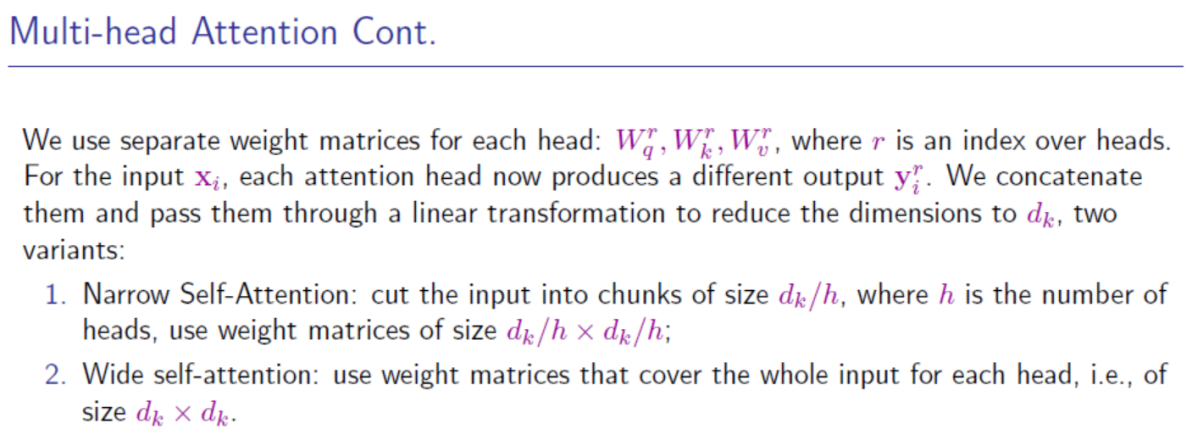
## Multi-Head Attention



把输入的向量分为多个头（等长的几份）



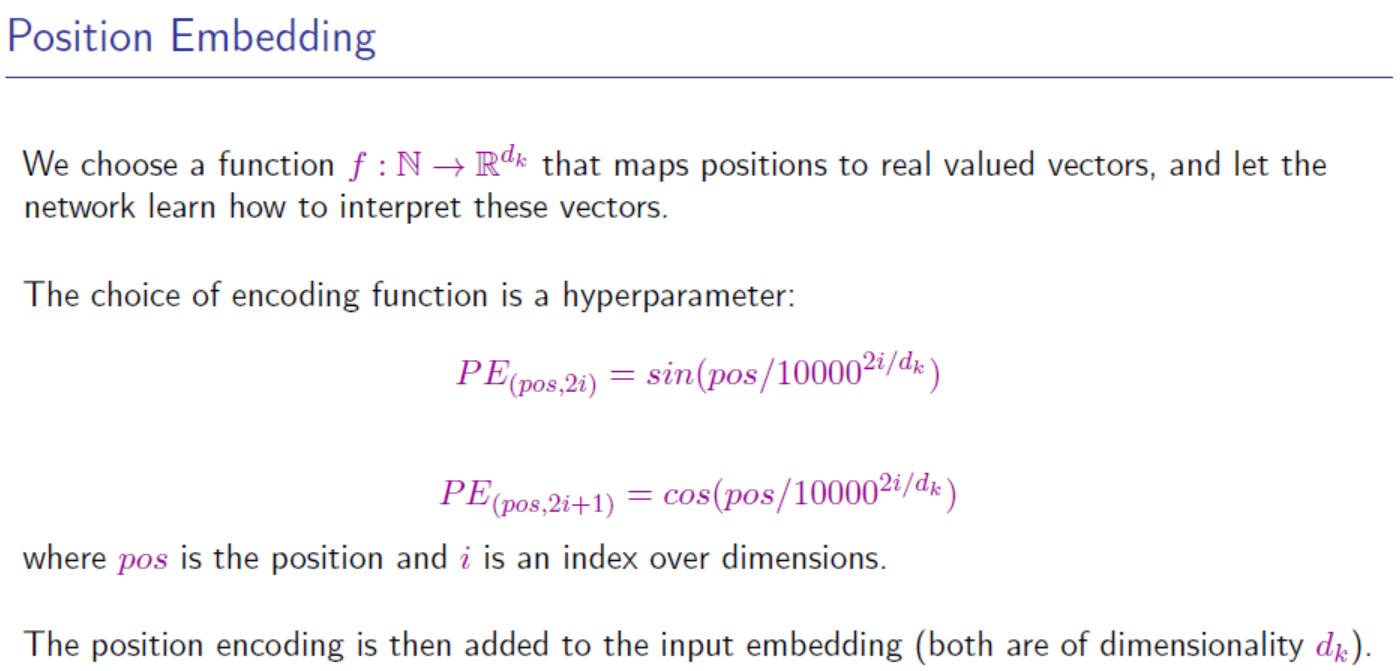




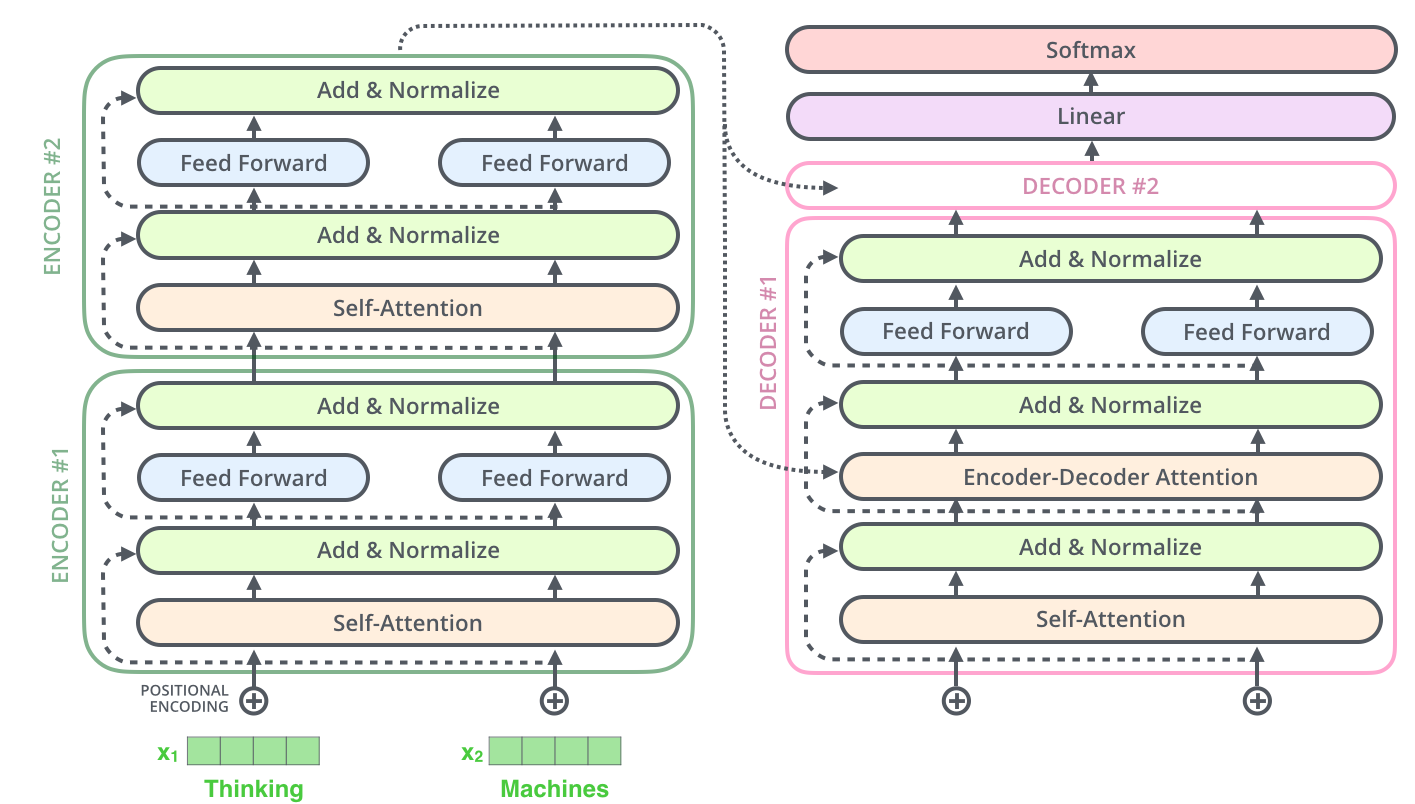
## Position Embedding（后面也会有改进）

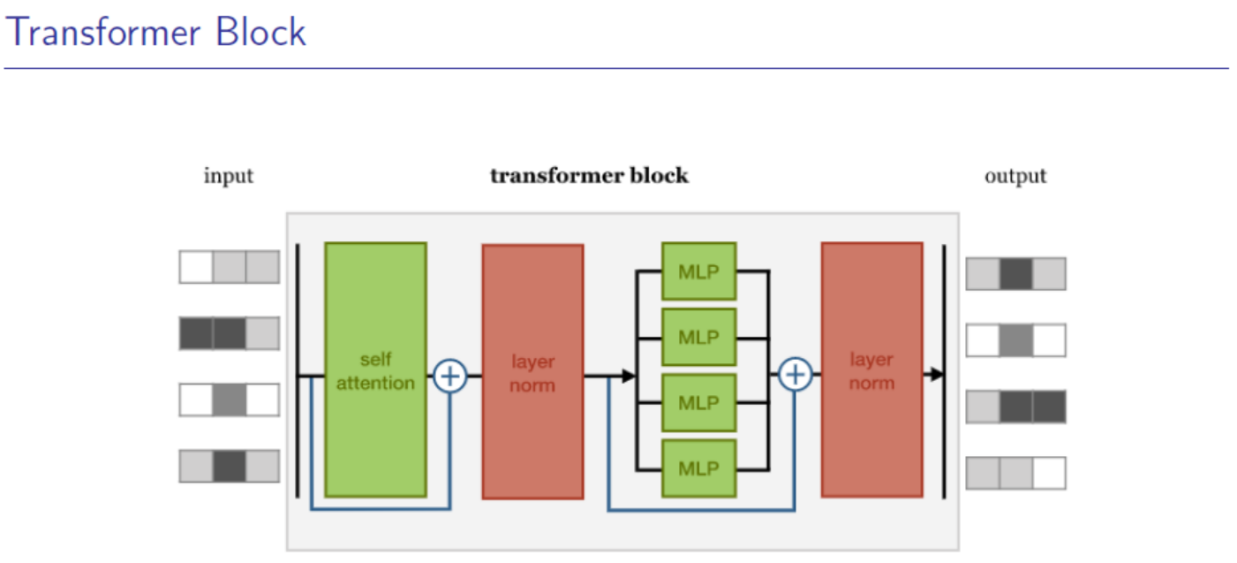
为什么要做position Embedding呢？

因为在做self-attention的时候，顺序是无关的，因此加多一个position



## Transformer



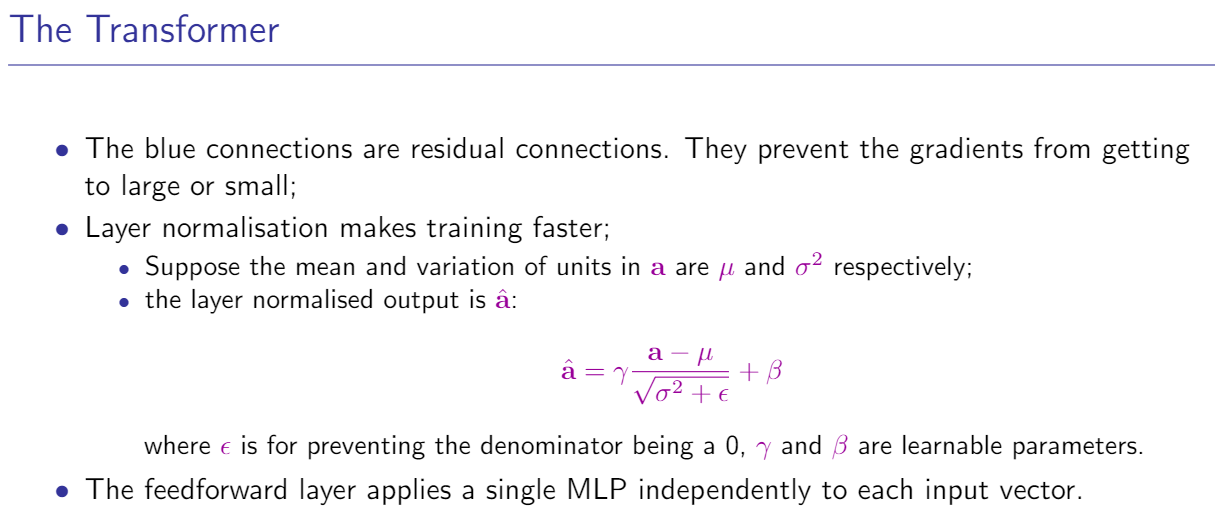


Self-attention == multi-head Self-attention

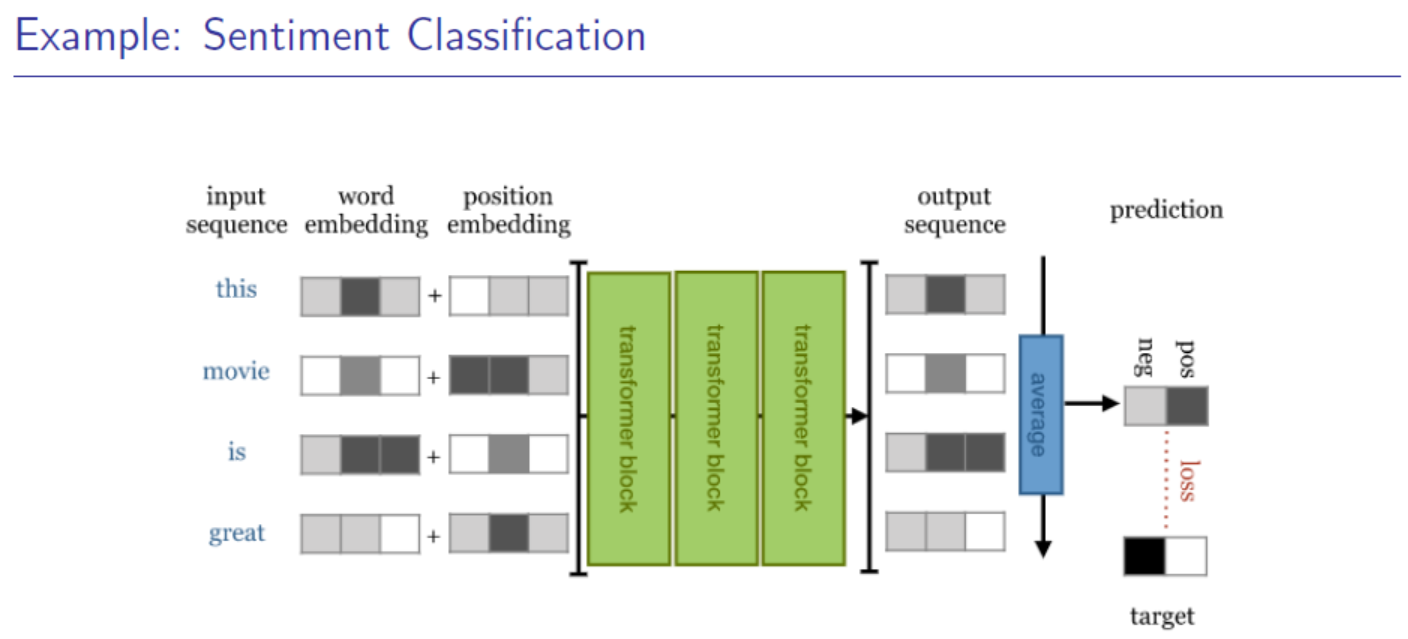
Norm = 正则化

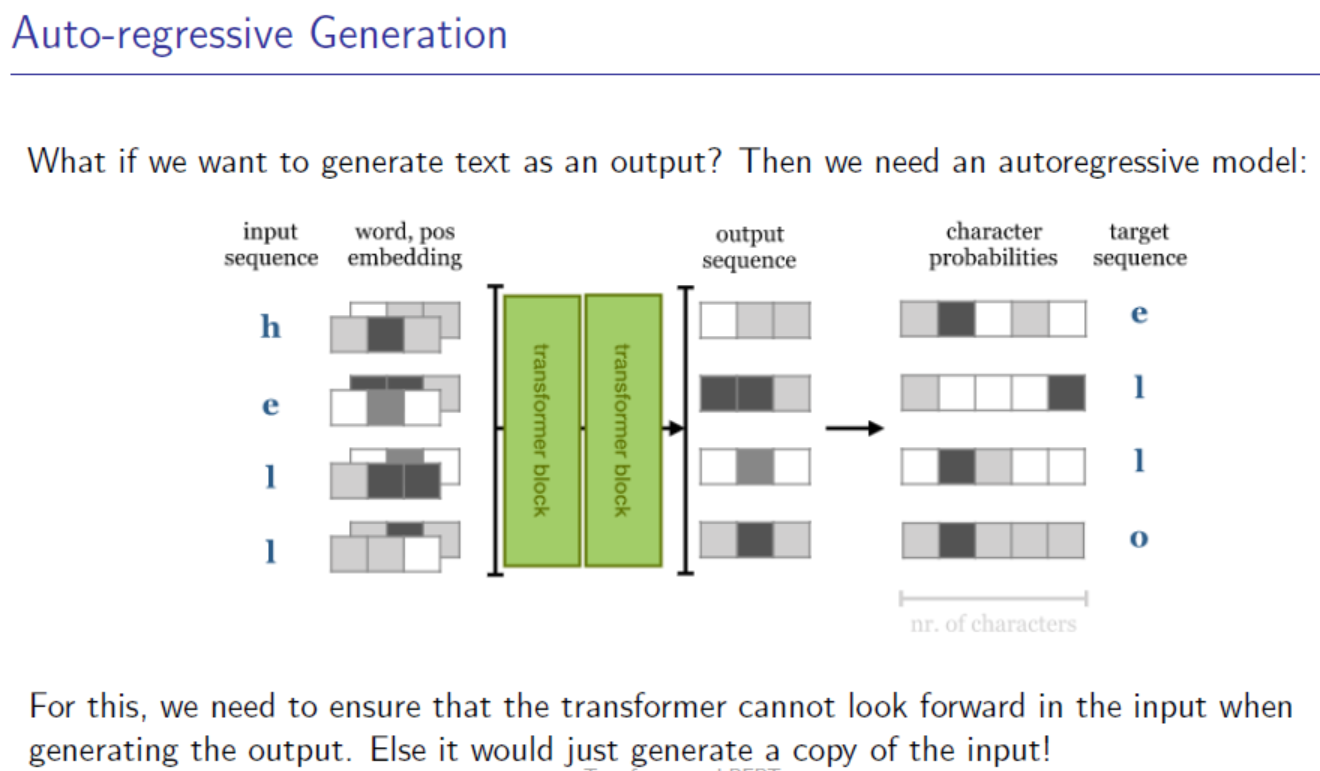
MLP：单层神经网络（relu激活）

蓝色：残差连接（训练更快+更稳定<--阻止梯度消失）

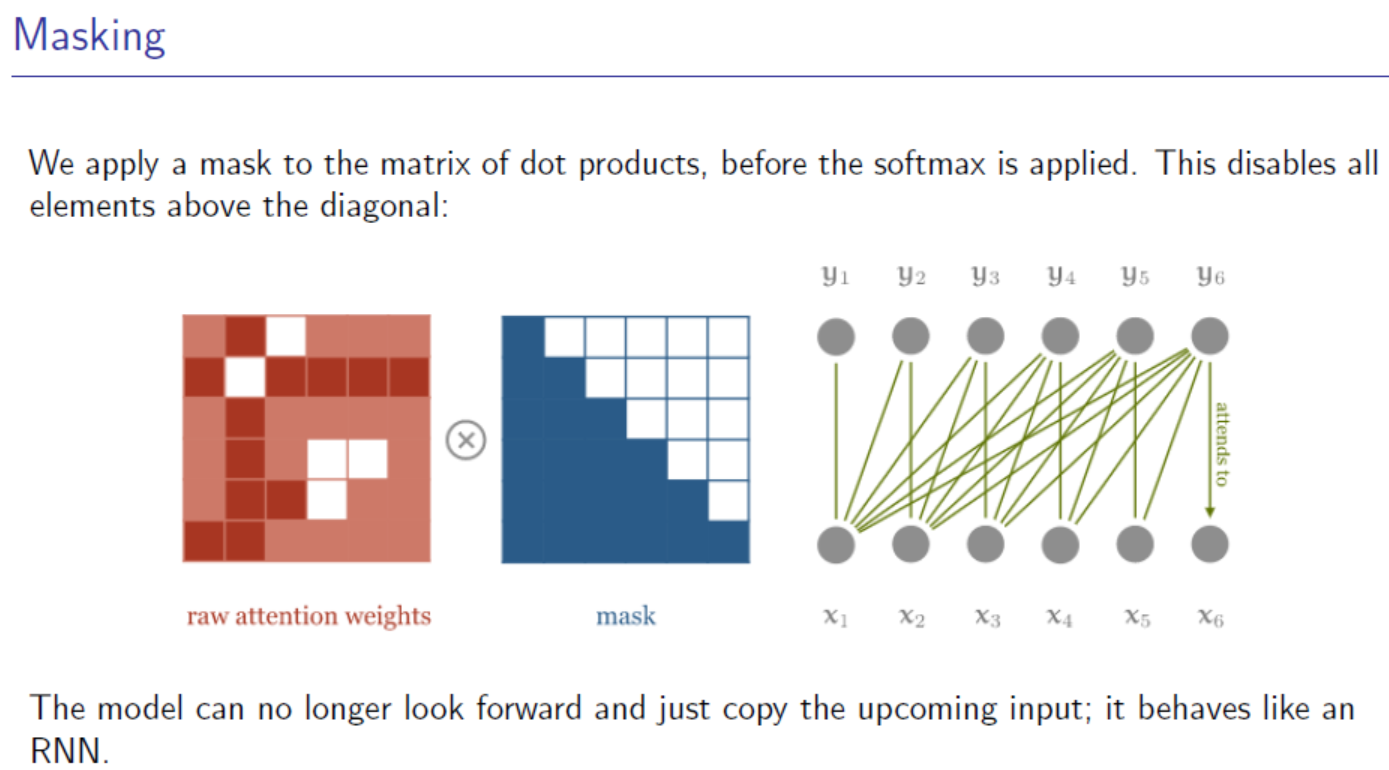


## Application





Mark：加权平均的时候不带入后面的数据x进行计算attention（防止偷窥）



<https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html>