需要 attention 的原因

最基本的 seq2seq 模型包含一个 encoder 和一个 decoder，通常的做法是将一个输入的句子编码成一个固定大小的 state，然后作为 decoder 的初始状态，但这样的一个状态对于 decoder 中的所有时刻都是一样的。

不同的部分重要的程度不同，所以我们需要对不同的部分施加不同的注意力。

对 attention 的理解

当我们人在看一样东西的时候，我们当前时刻关注的一定是我们当前正在看的这样东西的某一地方，换句话说，当我们目光移到别处时，注意力随着目光的移动野在转移，这意味着，当人们注意到某个目标或某个场景时，该目标内部以及该场景内每一处空间位置上的注意力分布是不一样的。

这一点在如下情形下同样成立：当我们试图描述一件事情，我们当前时刻说到的单词和句子和正在描述的该事情的对应某个片段最先关，而其他部分随着描述的进行，相关性也在不断地改变。从上述两种情形，读者可以看出，对于 Attention 的作用角度出发，我们就可以从两个角度来分类 Attention 种类：空间注意力和时间注意力。

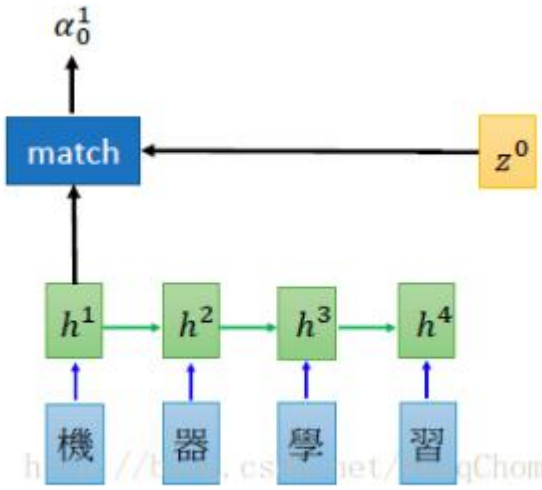
加入 attention 的原因

序列输入时，随着序列的不断增长，原始根据时间步的方式的表现越来越差，这是由于原始的这种时间步模型设计的结构有缺陷，即所有的上下文输入信息都被限制到固定长度，整个模型的能力都同样收到限制，我们暂且把这种原始的模型称为简单的编解码器模型。编解码器的结构无法解释，也就导致了其无法设计。

Attention-based Model

Attention-based Model 其实就是一个相似性的度量，当前的输入与目标状态越相似，那么在当前的输入的权重就会越大，说明当前的输出越依赖于当前的输入。Attention 并算不上是一种新的 model，而仅仅是在以往的模型中加入 attention 的思想。

Attention 机制



attention 其实就是一个当前的输入与输出的匹配度。在上图中，即为 h1h1 和 z0z0 的匹配度，h1h1 为当前时刻 RNN 的隐层输出向量，而不是原始输入的词向量，z0z0 初始化向量，如 rnn 中的 initial memory，其中的 match 为计算这两个向量的匹配度的模块，出来的 α10α01 即为由 match 算出来的相似度。这个就是 attention-based model 的 attention 部分。

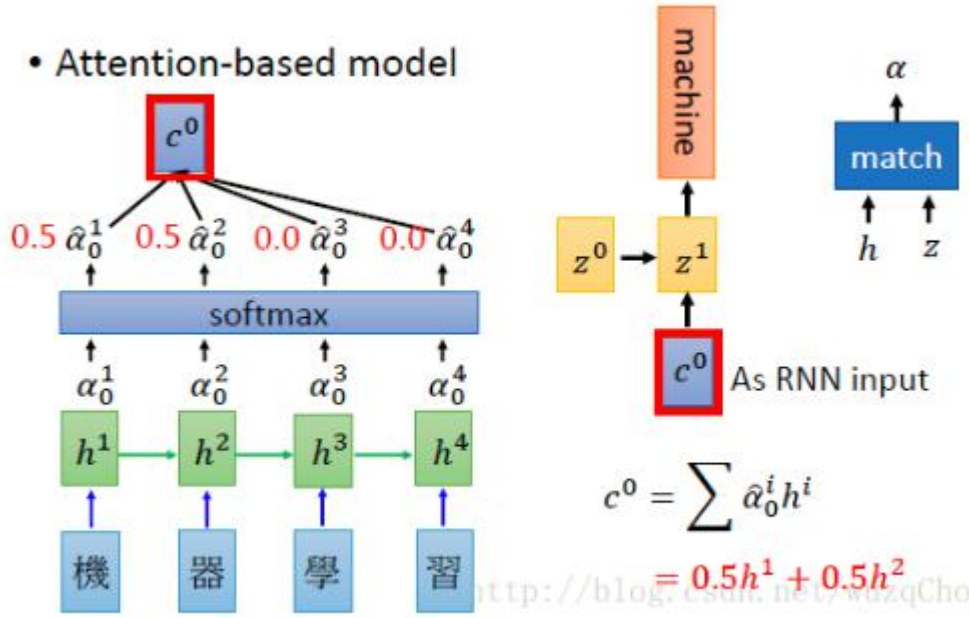
match 理论上可以通过任何计算两个向量的相似度的方法计算，比如：

余弦相似度

一个简单的 神经网络，输入为 hh 和 ww，输出为 α α

或者矩阵变换 α=hTWz α=hTWz

现在我们已经由 match 模块算出了当前输入输出的匹配度，然后我们需要计算当前的输出（实际为 decoder 端的隐状态）和每一个输入做一次 match 计算，分别可以得到当前的输出和所有输入的匹配度，由于计算出来并没有归一化，所以我们使用 softmax，使其输出时所有权重之和为 1。那么和每一个输入的权重都有了，那么我们可以计算出其加权向量和，作为下一次的输入。



那么再算出了 c^0 之后，我们就把这个向量作为 rnn 的输入，然后 d 第一个时间点的输出的编码 z^1 由 c^0 和初始状态 z^0 共同决定。我们计算得到 z^1 之后，替换之前的 z^0 再和每一个输入的 encoder 的 vector 计算匹配度，然后 softmax，计算向量加权，作为第二时刻的输入……如此循环直至结束。

$$u_t^i = v^T tanh(W_1 h^i + W_2 Z^t)$$

$$\alpha_t^i = softmax(u_t^i)$$

$$c^t = \sum_i \alpha_t^i h^i$$

得到 c^t 之后，就可以作为第 t 时刻 RNN 的 input，而 Z^t 可以作为 t 时刻 RNN 的隐状态的输入，这样就能够得到新的隐状态 Z^{t+1}，如此循环，直到遇到停止符为止。