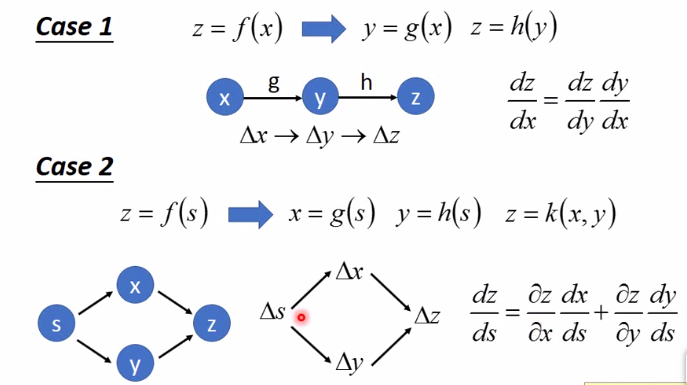
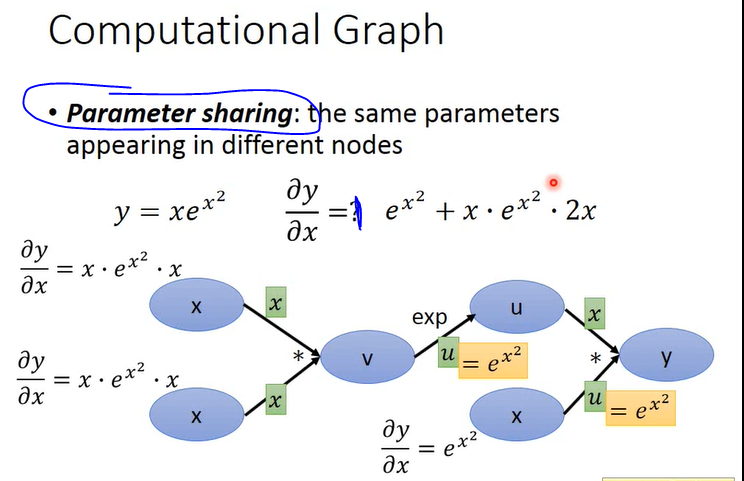
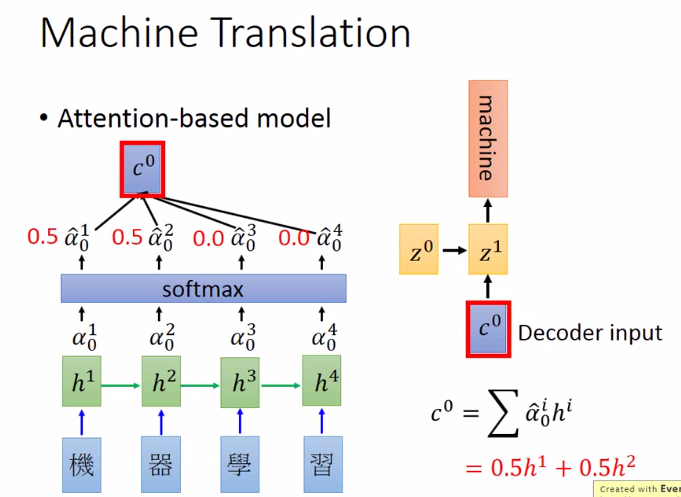
# 1.反向传播



参数共享时的情况:



# 2.Attention注意力模型



h1是encode阶段某一时刻的输出

z0是一个初始化的向量，也可以是可训练的

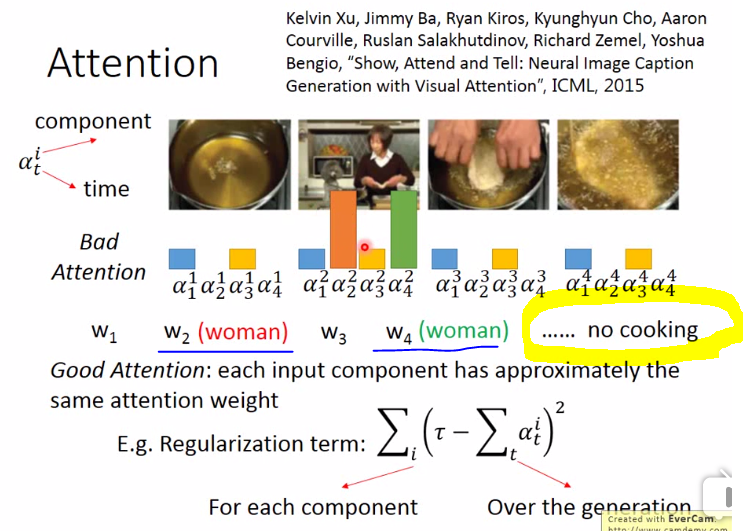
α是h1与z0之间计算匹配度的匹配结果，这个计算匹配度的方法可以有很多种，自己设计

然后进过softmax，将结果与h1相乘，然后累加得到c0，将z0和c0作为decoder阶段的输入。然后重复循环。

优点:计算匹配度的时候，比如考虑机器的时候，会忽略掉学习，将注意力集中在机器上，而计算学习时可以忘掉机器。因为当h1是机的时候，机与器的匹配度α是很高的，而与学习无关，这样会反应在c上。

# 3.训练中存在的一些技巧

## 1.注意力模型技巧：



例子1：生成的句子里出现好几个woman这个单词，是因为attention的注意力过多的放在了女人上面，因此一个好的注意力模型应该在不同的图片里给予相同的注意力。

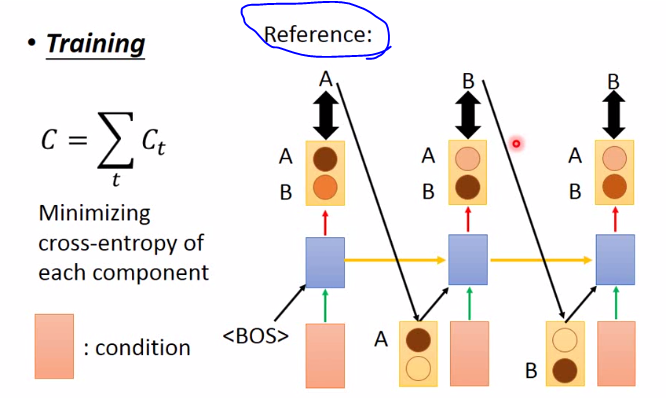
解决办法举例:添加正则化项，τ是每张图片注意力(α匹配度)的累加，用τ减去本章图片的注意力(α匹配度),注意力小的，得到的值就大，损失就越大，这样就会对没有注意力的更加注意

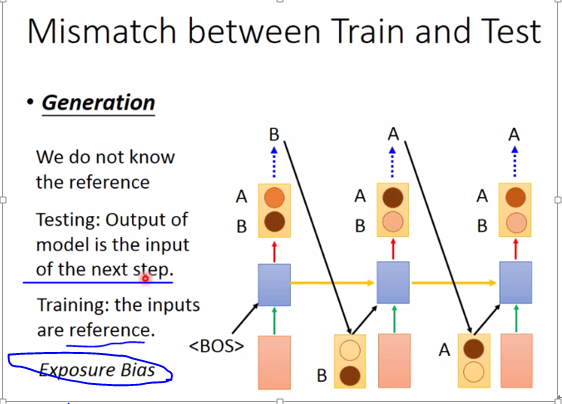
## 2.RNN技巧：

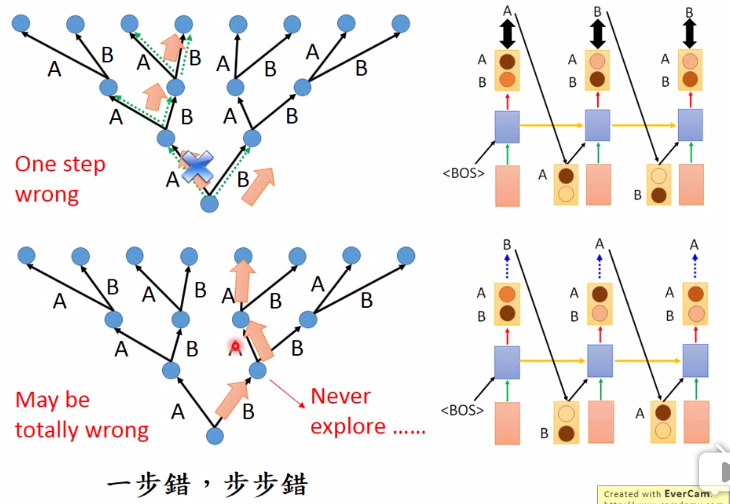
在训练的时候，我们是拿正确的结果A作为下一步的输入，而不是预测值。

而在测试的时候，我们是拿预测值作为下一步的输入，这样存在偏差，称为Exposure Bias(曝光偏差)。

这样导致的后果就是图3中，如果训练时存在错误预测的是B，但是我们下一步的输入仍然是正确值A，但是假如用一样的句子来测试时就会将错误的值B用做下一步的输入，这样以来，往B走的路线是从来没有学习过的，就会出现完全错误的句子。

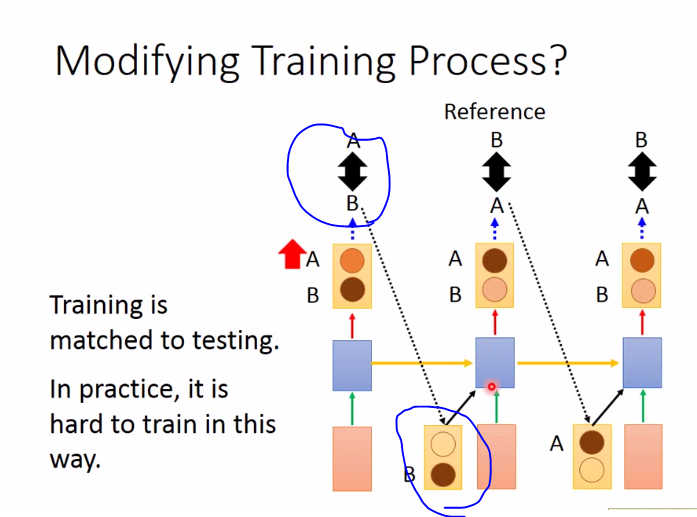






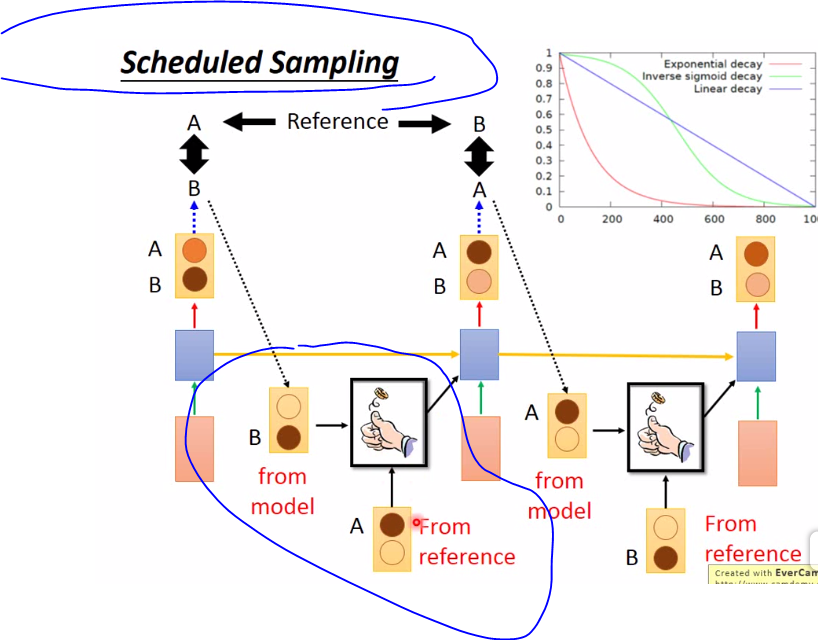
对于以上问题，那么假如我们修改训练过程，使用预测值作为下一步的输入的话，你会发现训练时很难loss收敛。

假如我们预测值是B，真实值是A，那么由于存在偏差，我们会调高A的权重。这个时候，我们把错误的B作为下一步的输入，但是当我们完全重新开始训练第二回的时候，由于前一次训练，将A的权重提高了，那么这个时候输出的结果变成A了，上一次是B，当把A作为下一步的输入的时候，你会发现上一次B作为输入的训练结果全部作废了甚至可能产生更加奇怪的结果。



解决办法：scheduled sampling(定期取样)，以一定的概率选择预测值或者是真实值。

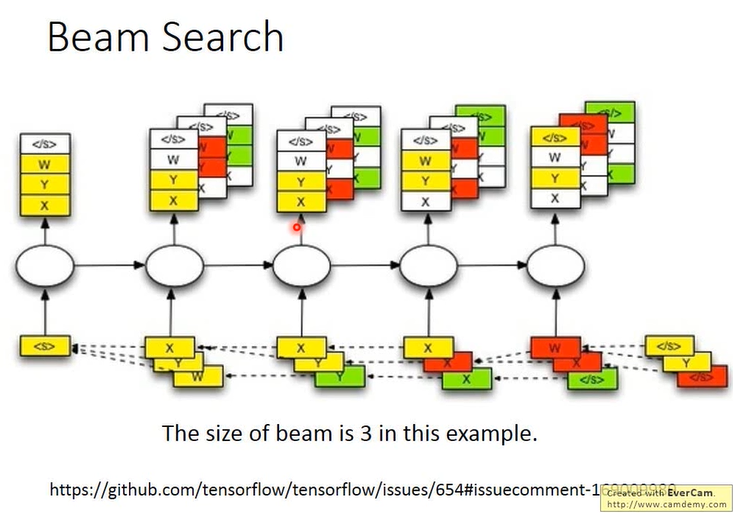
训练刚开始的时候较大概率选择真实值，然后慢慢下降，逐渐选择预测值



Grid Search：暴力搜索，计算所有情况，找到最大的情况

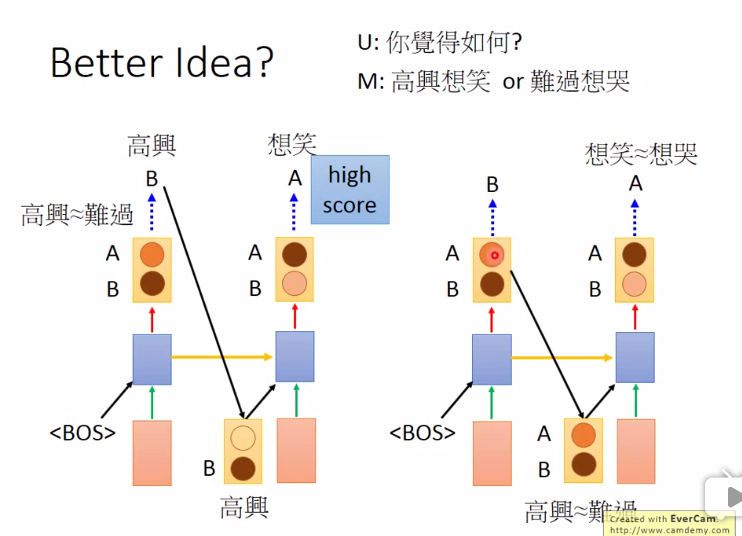
Beam Search：每次只选择当前几条概率最高的

其它 Search：每次只选择当前概率最高的，依次找下去.这种简单粗暴，但是问题也明显，0.6->0.6->0.6,明显没有0.4->0.9->0.9的组合概率高。



还有一种思路：是将分布的权重W直接作为下一步的输入，不使用预测值，也不使用实际值。这样做的好处是，第一，不用做scheduled sampling，所有的信息都传递给下一步。第二，维度没有变化，不用做处理，直接就可以拿来用。但是同时也存在问题，比如，高兴和难过的概率差不多，你都作为下一步的输入，那么输出想笑和想哭的概率也差不多，最后得到的结果有可能是高兴想哭。而之前的直接输入高兴，下一步就只会输出想笑。

因此，scheduled sampling效果更好



如果是按照一个个单词是否匹配作为损失函数，那么the dog is is fast其实损失值已经很低了，很难再近一步。但是它的语法其实是完全不可用的。如果存在一种损失函数R，是基于整个句子的，那么the dog is is fast其实损失较大。但是问题是，这样的损失函数是不可以微分的，梯度无法下降，因为当你调参时，不同单词的分布会有变化，但是概率最大的仍然是那个词，导致最后的句子y没有变化，那么R的结果也不会有变化。

