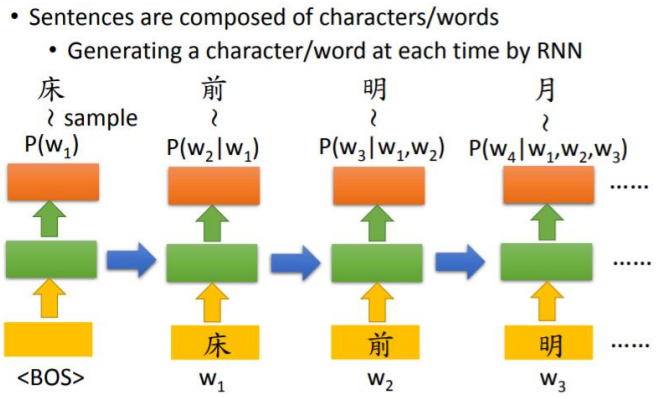
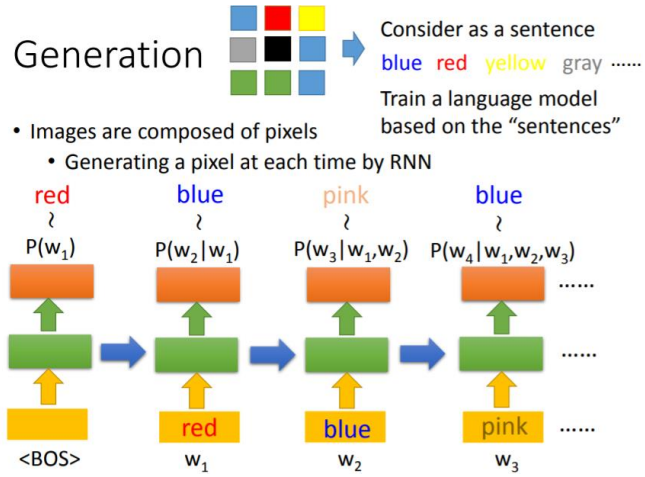
Sequence to Sequence

Generation

可以 generation 文字，也可以 generation 图片

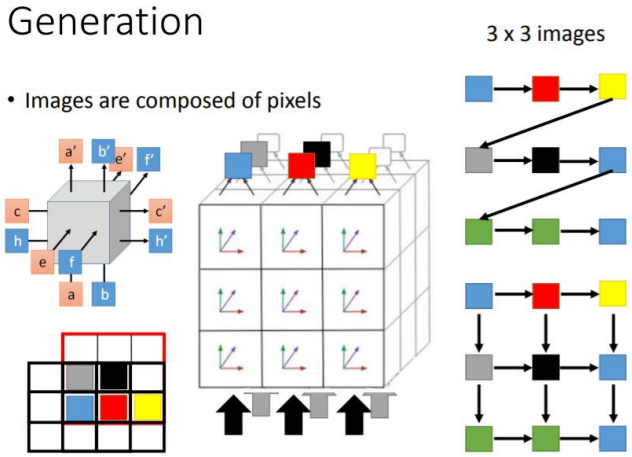


如果想生成一张图片，我们可以把图片的每个 pixel 看成一个 character，从而组成一个 sentence，比如下图中蓝色 pixel 就表示 blue，红色的 pixel 表示 red……



对于一般的 generation，如下图的右上部所示，先根据蓝色 pixel 生成红色 pixel，再根据红色 pixel 生成黄色 pixel，再根据黄色 pixel 生成灰色 pixel，……,并没有考虑 pixel 之间的位置关系

还有另外一个比较理想的生成图像方法，如果考虑了 pixel 之间的位置关系，如下图的右下部分所示，黑色 pixel 由附近的红色和灰色 pixel 生成



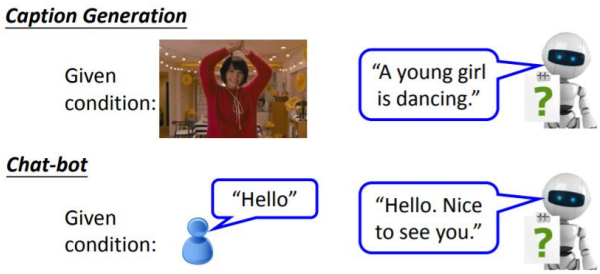
在上图的左上角部分，该 LSTM 块输入了三组参数，把这些方块叠起来就可以达到中部位置图像的效果

对于一个 3*3 的画布，就是哦们想要生成的图像的大小，一开始画布上还没有任何内容；现在在画布的作下角放入一个 convolution 的 filter，把器输出再放到 3D 的 LSTM 的左下角(input),由于 LSTM 也是有 memory 的，经过不断的 process，可以到左上角，从而产生蓝色的 pixel，再把蓝色的 pixel 放到画布的左下角

Conditional Generation

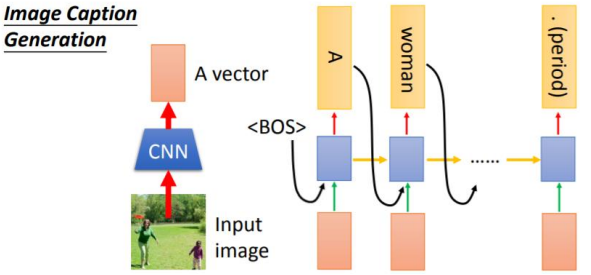
必须是有条件的 generation，要联系上下文

- We don't want to simply generate some random sentences.
- Generate sentences based on conditions:



如果要对一张图片进行解释，我们先使用一个 CNN model 将 image 转化为一个 vector(红色方框)，先生成第一个单词 "A"，在生产第二个单词时，还需要将整个 image 的 vector 作为输入，不然 RNN 可能会忘记 image 的一些信息

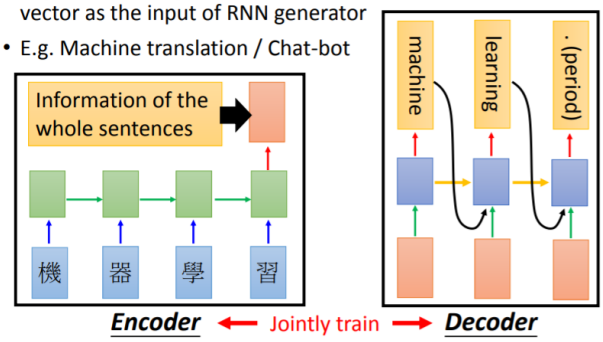
- Represent the input condition as a vector, and consider the vector as the input of RNN generator



RNN 也可以用来机器翻译，比如我们想要翻译 "机器学习"，就先把这个四个字分别输入绿色的 RNN 里面，最后一个时间节点的输出(红色方框)，就包括了整个 sentence 的 information，这个过程称之为 Encoder

把 encoder 的 information 再作为另外一个 RNN 的 input，再进行 output，这个过程称之为 Decoder，encode 和 decode 是 jointly train 的，这两者的参数可以是一样的，也可以是不一样的

- Represent the input condition as a vector, and consider the vector as the input of RNN generator
- E.g. Machine translation / Chat-bot

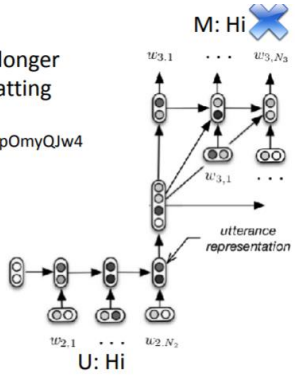


Encoder 将之前所有说过的话都进行整合了，再作为 decoder 的 input

M: Hello
U: Hi
M: Hi

Need to consider longer context during chatting

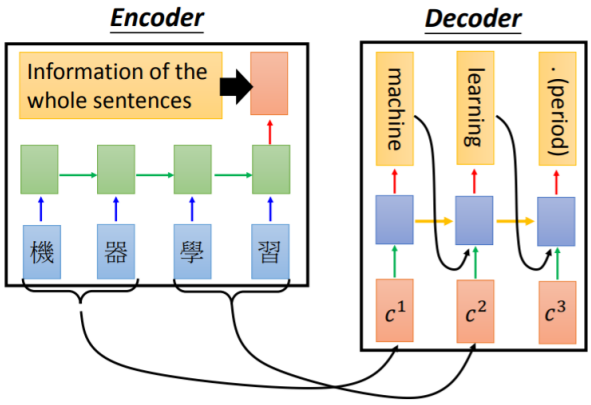
https://www.youtube.com/watch?v=e2MpOmyQJw4



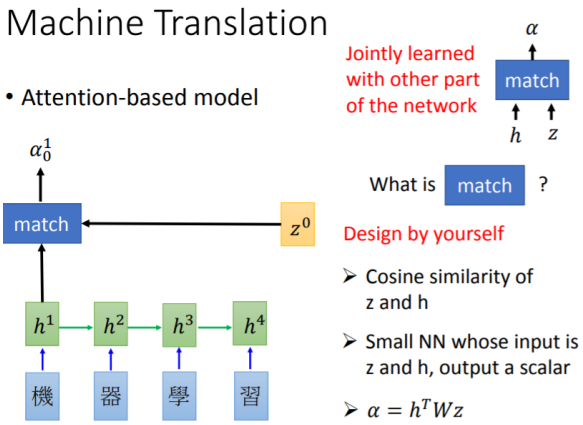
Attention

Dynamic Conditional Generation

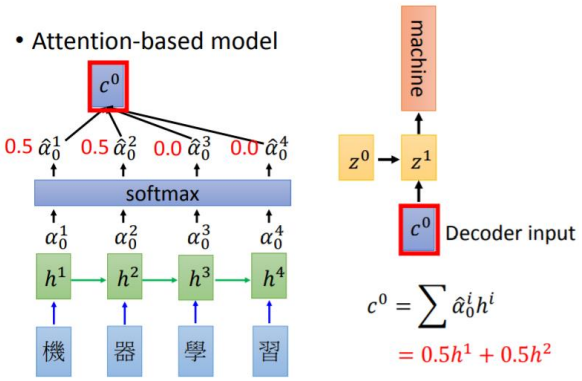
如果需要翻译的文字非常复杂，无法用一个 vector 来表示，就算可以表示也没办法表示全部的关键信息，这时候如果 decoder 每次 input 的都还是同一个 vector，就不会得到很好的结果
因此我们现在先把“机器”作为 decoder 的第一个输入c^1，这样就可以得到更好的结果



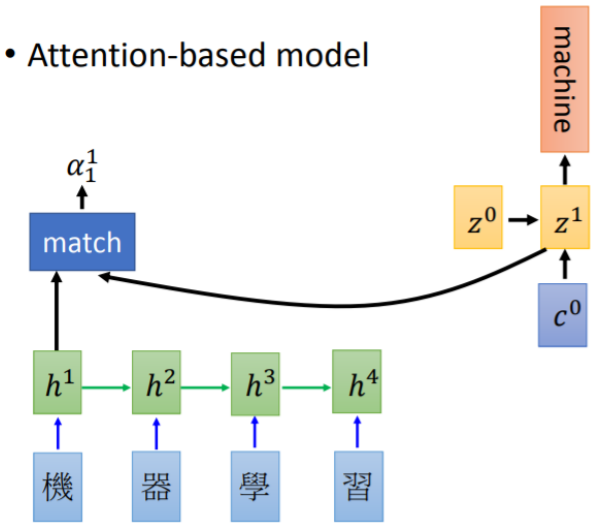
Machine Translation
绿色方块为 RNN 中 hidden layer 的 output，分别是h^1, h^2, h^3, h^4;还有一个 vector z^0，是可以通过 network 学出来的；把这两者输入一个 match 函数，可以得到h^1, z^0之间的 match 分数α_0^1
这个 match 函数可以自己设计，可以有参数，也可以没有参数



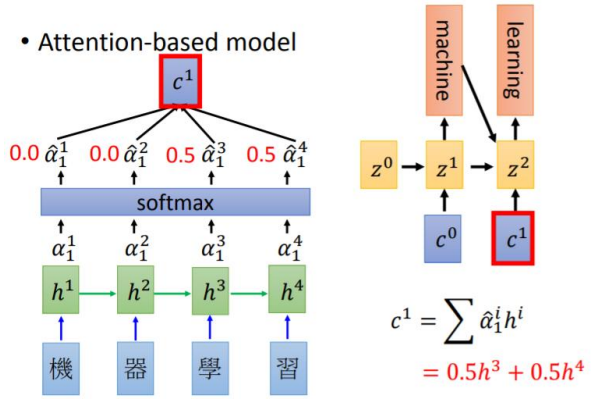
得出的$\alpha_0^1, \alpha_0^2, \alpha_0^3, \alpha_0^4$再输入 softmax 函数，得到$\hat{\alpha}_0^1, \hat{\alpha}_0^2, \hat{\alpha}_0^3, \hat{\alpha}_0^4$，乘上$h^i$再加起来，就可以得到$c^0$,来作为 decoder input，根据 decoder，可得出第一个翻译的单词“machine”；z^1为 decoder RNN 其中 hidden layer 的输出，用z^1再来得出相应的 decoder input



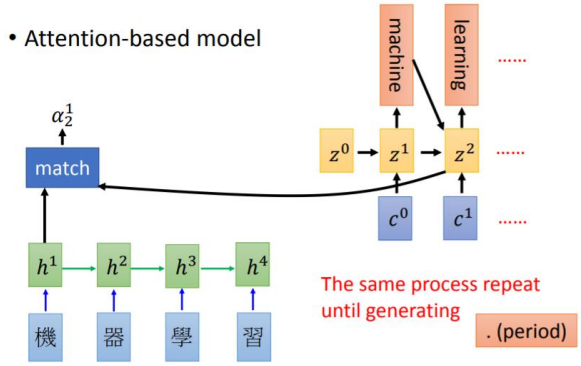
再来计算z^1, h^1之间的 match 分数α_1^1，



再继续算出c^1作为 decoder 的 input，从而得出第二个单词 “learning”

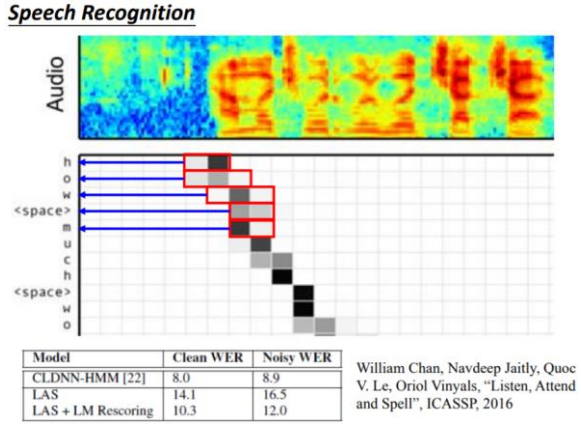


一直重复这个过程，只到全部翻译完成



Speech Recognition

语音信号也可以看成一系列的 vector sequence，第一个红色方框所在的帧 match 分数很高，就把对应的 vector 放到 decoder，machine 就会得出 “h”

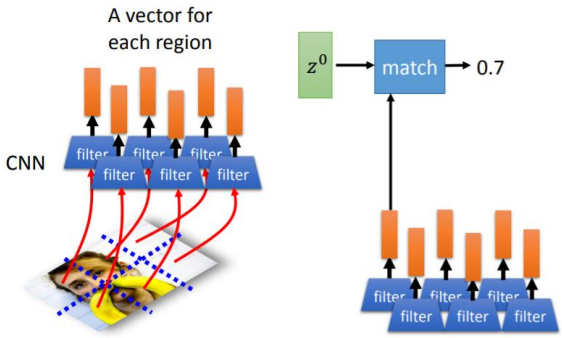


Model	Clean WER	Noisy WER
CLDNN-HMM [22]	8.0	8.9
LAS	14.1	16.5
LAS + LM Rescoring	10.3	12.0

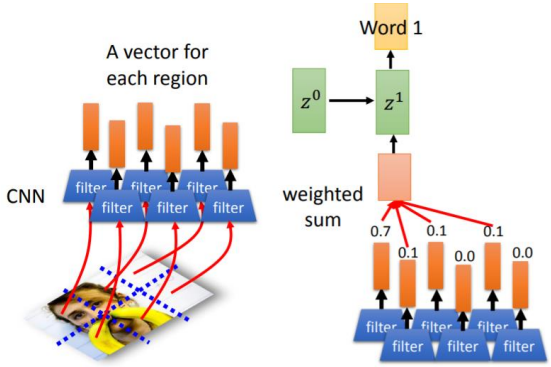
William Chan, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals, “Listen, Attend and Spell”, ICASSP, 2016

Image Caption Generation

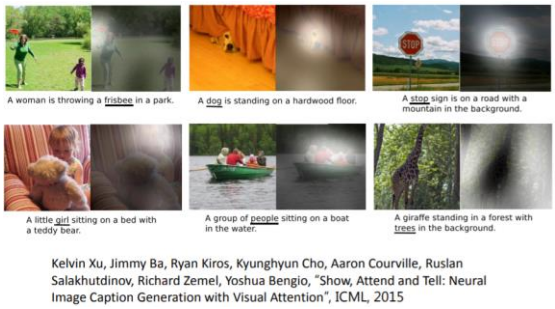
Image 可以分成多个 region，每个 region 可以用一个 vector 来表示，计算这些 vector 和z^0之间 match 的分数，为 0.7、0.1….，再进行 weighted sum，得到红色方框的结果，再输入 RNN，得到 Word1，此时 hidden layer 的输出为z^1



再计算z^1和 vector 之间的 match 分数，把分数作为 RNN 的 input，从而得出 Word2

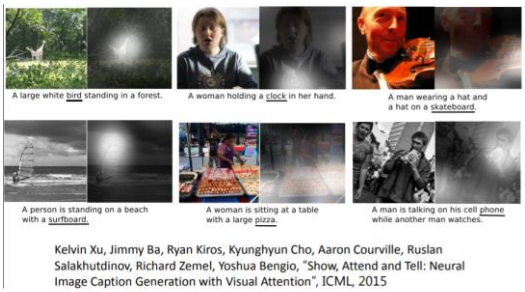


这里是一些具体的例子



Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015

但也会产生一些不好的结果

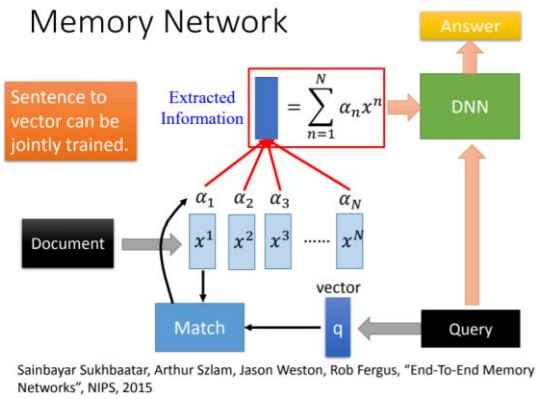


Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML, 2015

Memory Network

memory network 是有一篇文章，问 machine 一个问题，machine 能够给出对应的答案

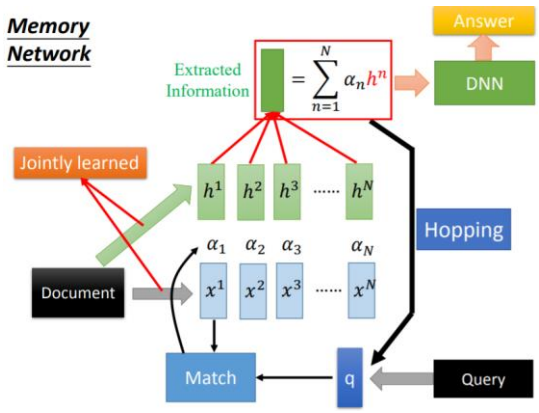
先将 document 分成多个 vector，再给出这些 vector 与问题 q 之间 match 的分数α^1 ……α^N,与x^i分别相乘，得到 extracted information，作为 DNN 的 input，再进行不断地训练，从而得到对应的 answer



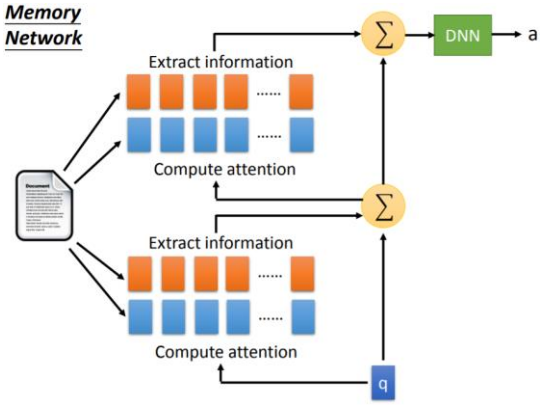
还有一个更加复杂的版本

Match 的部分和 extracted 的部分不一定是一样的，相当于输入了两组不同的参数，比如可以把x^i乘上一个 matrix，就可以变成另外一组参数h^i,这个 matrix 是可以自动学出来的

这时计算 extracted information 的方式就发生了一些变化；sum 之后的值不仅输入 DNN，还会与问题 q 进行结合，再重新计算 match 分数，不断进行一个反复思考的过程



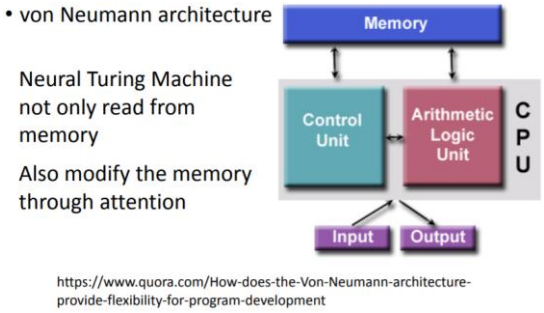
q 可以用来计算 attention，计算出 extracted information 之后，与 q 加起来；上一次 sum 的结果会再继续参与计算 attention，在 extract，再进行 sum，作为 DNN 的 input，得到最后的答案



可以把这个模型看作是两层 layer 的 network，也有 back propagation 来更新 network 的参数

Neural Turing Machine

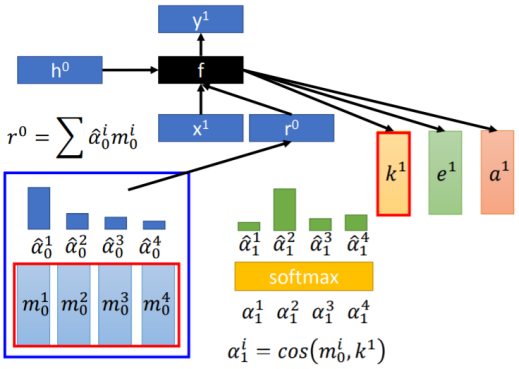
Neural Turing Machine 不仅可以读取 memory 的内容，也可以根据 match 分数来修改 memory 的内容；即你不仅可以通过 memory 读取 information，也可以把 information 写到 memory 里面去



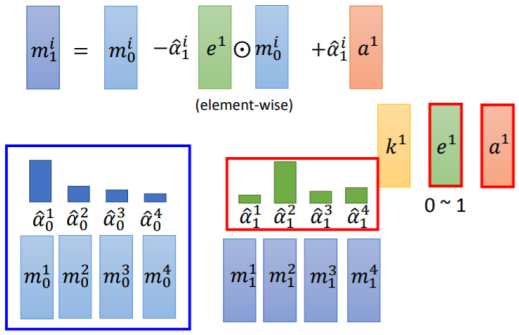
对于初始值的 memory m_0^i,计算出对应的 match 分数$\hat{\alpha}_0^i$，在进行 weighted sum，得出r^0，来作为另外一个 network f 的 input，这个 network 相当于是一个 controller，f 可以是 DNN、SVM、GRU 等

r^0, x^1作为该 network 的输入，可以得出三个 output，这些 output 可以控 memory，比如新的 attention 是什么，新的 memory 里面的值如何进行修改

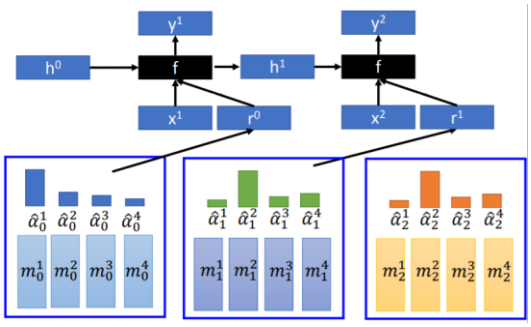
新的 attention 计算加入k^1，即$\alpha_1^i = \cos(m_0^i, k^1)$,再输入 softmax，得到新的$\hat{\alpha}_1^i$的 distribution



e^1里面的值表示把原来 memory 里面的值清空，a^1表示把新的值写入 memory，通过下面的公式再计算新的 memory m_1^i



得到新的 memory m_1^i之后，再计算出新的 attention $\hat{\alpha}_1^i$，再进行 weighted sum 得到r^1，加上新的x^2作为 network 的 input，得出新的 output，再来对整个 network 进行操控

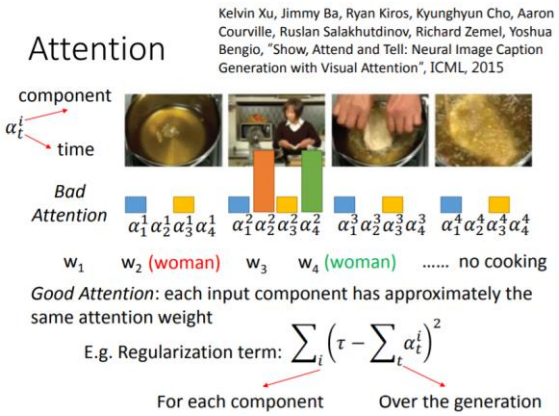


Tips for Generation

Attention

α_t^i下标表示 time，上标表示 component，该视频包括 4 个 component，有 4 个时间节点；在第一个时间节点产生 attention $\alpha_1^1, \alpha_1^2, \alpha_1^3, \alpha_1^4$,生成了第一个 word w_1

attention 也是可以调节的，有时会出现一些 bad attention，在产生第二个 word($w_2, women$)时，focus 到第二个 component，在产生第四个 word w_4时，也 focus 到第二个 component 上，也是 women，多次 attention 在同一个 frame 上，就会产生一些很奇怪的结果

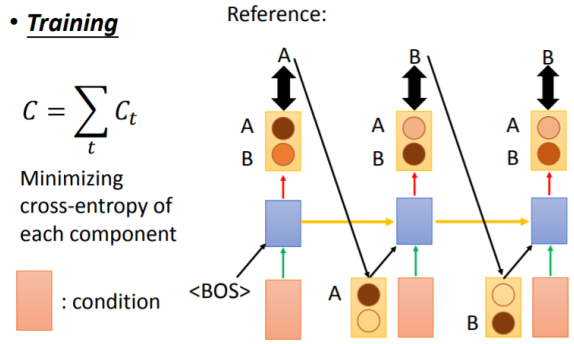


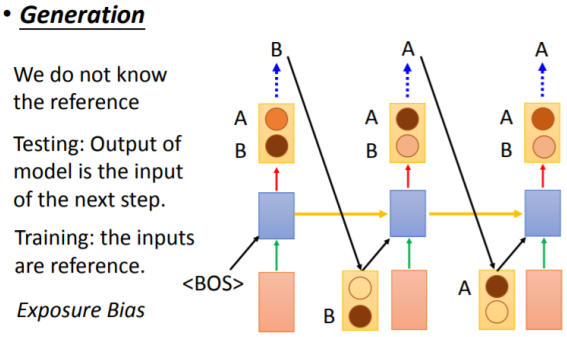
Good Attention: 至少要包含 input 的每一个 frame，每个 frame 都应该 attention 一下，每个 frame 进行 attention 的量也不能太多，这个量最好是同等级的

Q: 那么如何保证这个标准呢?

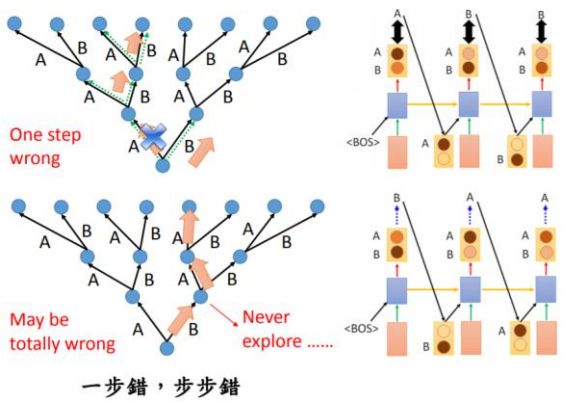
A: 有学者提出了一个 regularization term，有一个新的可学习的参数τ，$\sum_t \alpha_t^i$表示所有 attention 的 sum，再计算$\sum_i\left(\tau-\sum_t \alpha_t^i\right)^2$的值，这个值越小越好；对于上文所说的 bad attention 的情况，这个值算出来是很大的，因此 network 就会再进行不断地学习，只到 term 的值不断减小

Mismatch between Train and Test





Exposure Bias: 在 training 的时候，input 为真正的答案；但在 testing 的时候，input 为上一个时间节点的 output 在下图中，由于 RNN 从来没有到右子树 B 训练过，但如果在 testing 时一开始就遇到了 B，network 就会出现错误



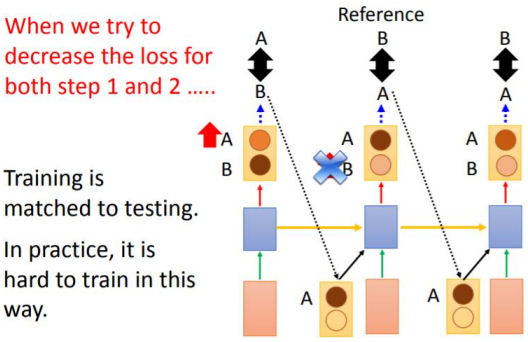
Modifying Training Process?

那么我们如何解决这种 mismatch 问题呢?可以尝试 modify 训练 process

如果 machine 现在 output B，即使是错误的 output(和 reference A 不一样)，我们也应该让这个错误的 output 作为下一次的 input，那么 training 和 testing 就是 match 的

但在实际操作中，training 是非常麻烦的；现在我们使用 gradient descent 来进行 training，第一个 gradient 的方向告诉我们要把 A 的几率增大，output B，在第二个时间点，input B,看到 reference 为 B，把 B 的几率增大，就可以和 reference 相对应起来；

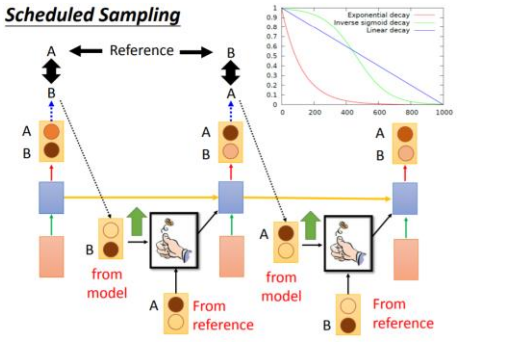
但实际上，第一个 output 为 A 的几率上升，那么 output 发生了变化，第二个时间点的 input 就发生了变化，是 A；那么我们之前学习到的让 B 上升就没有意义了



Scheduled Sampling

由于使用 Modify Training Process 很难 train，现在我们就使用 Scheduled Sampling 对于到底是 model 里的 output，如果是反面，就用 reference

右上角的图，纵轴表示 from reference 的几率，一开始只看 reference，reference 的几率不断变小，model 的几率不断增加

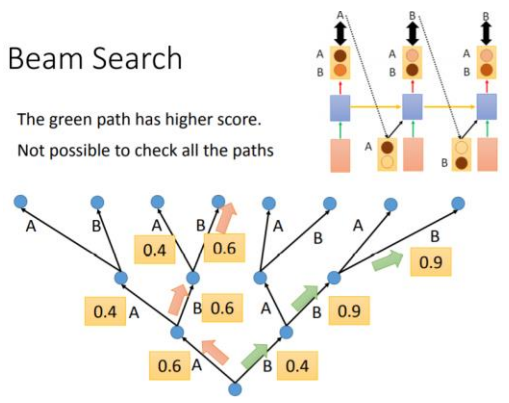


• Caption generation on MSCOCO

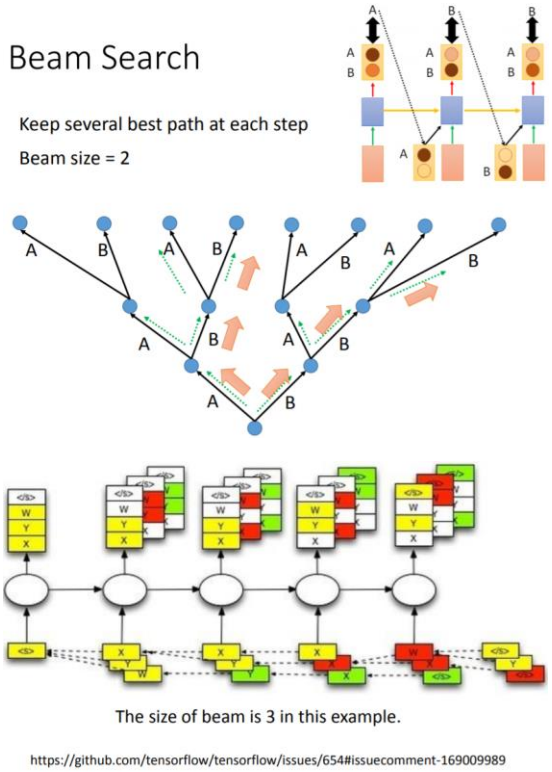
	BLEU-4	METEOR	CIDER
Always from reference	28.8	24.2	89.5
Always from model	11.2	15.7	49.7
Scheduled Sampling	30.6	24.3	92.1

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer, Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, arXiv preprint, 2015

Beam Search



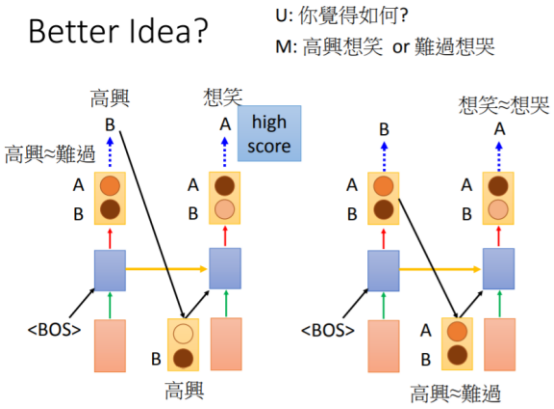
同时走两条 path

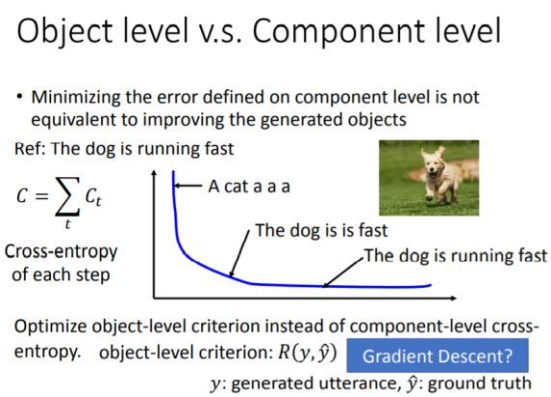


Better idea?

为什么不直接把 distribution 作为下一个 input 呢?即下图中的右图

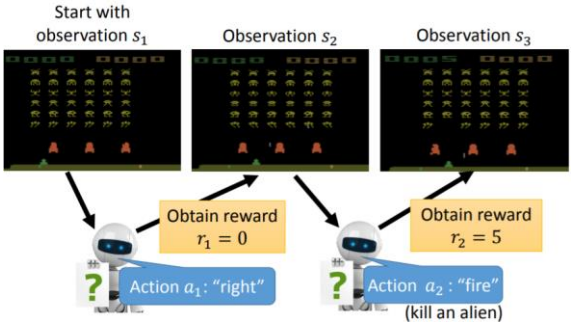
但右边的结果会比较差，对于几率接近的输入，左图会 sample 一个几率最高的（高兴），但右图会保留这个 distribution

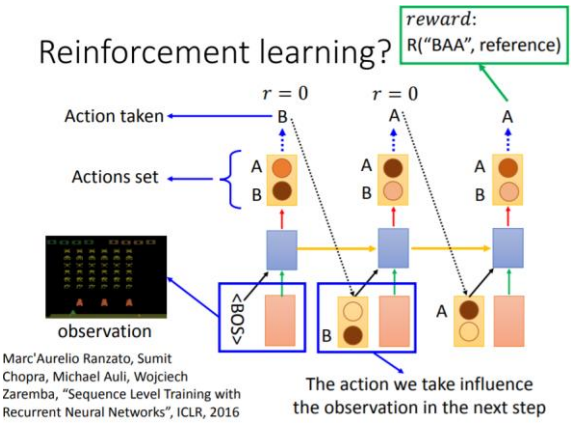




应该是考虑整个 sentence 的准确率如何，而不应该只考虑单个的 word

Reinforcement learning?





Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, Wojciech Zaremba, "Sequence Level Training with Recurrent Neural Networks", ICLR, 2016