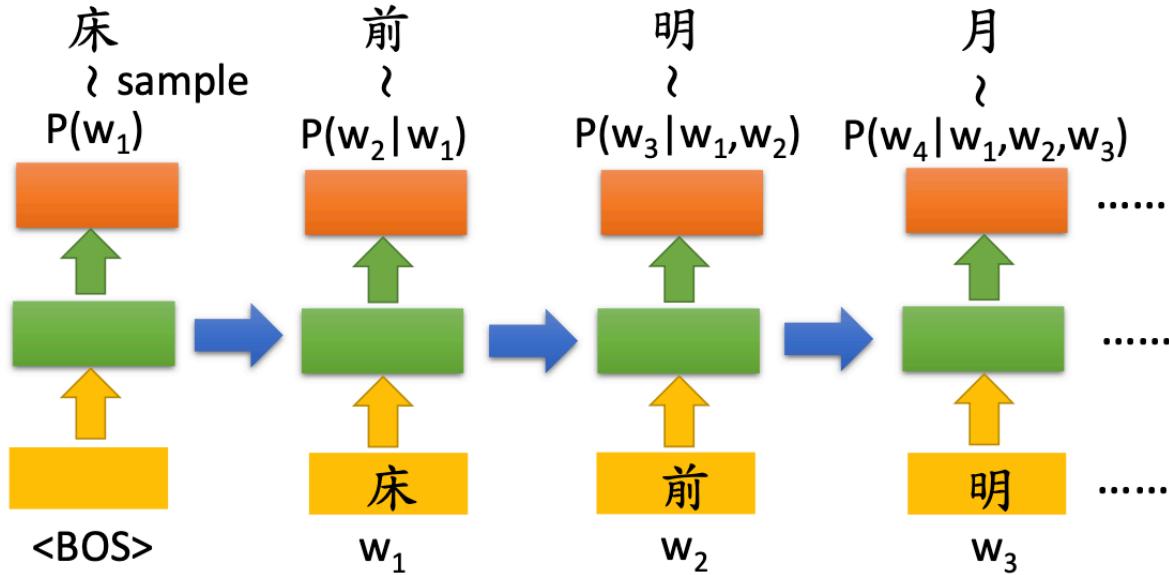


## Generation

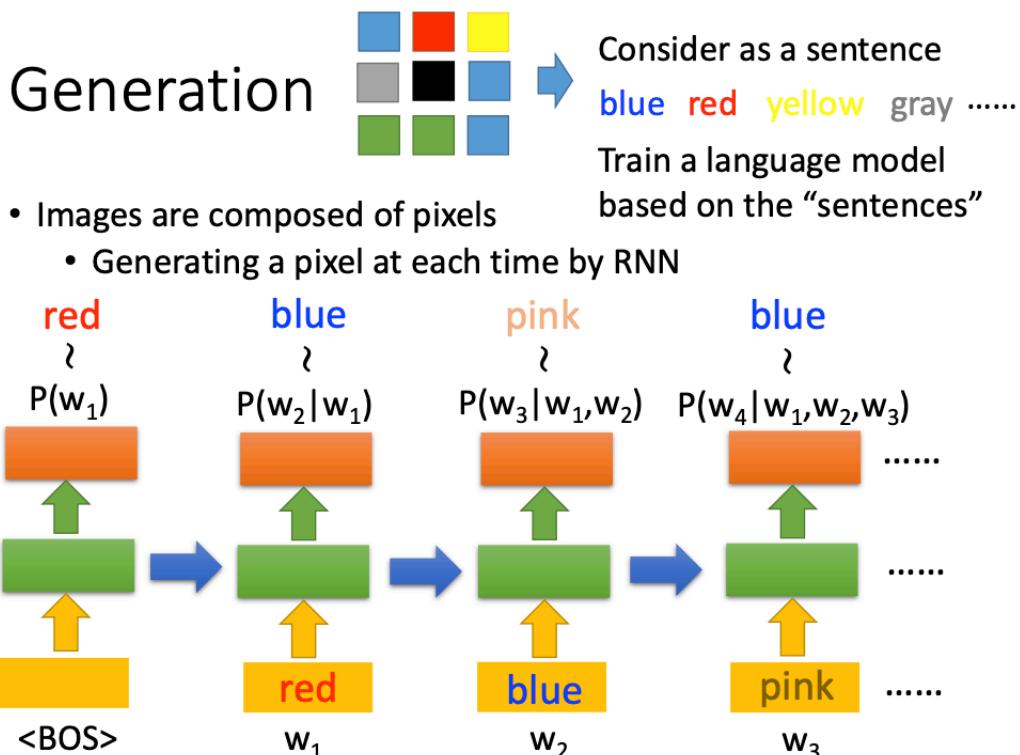
可以generation文字，也可以generation图片

- Sentences are composed of characters/words
  - Generating a character/word at each time by RNN



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,.....

接下来就可以用language model来生成图片，最初时间步的输入是特殊字符BOS

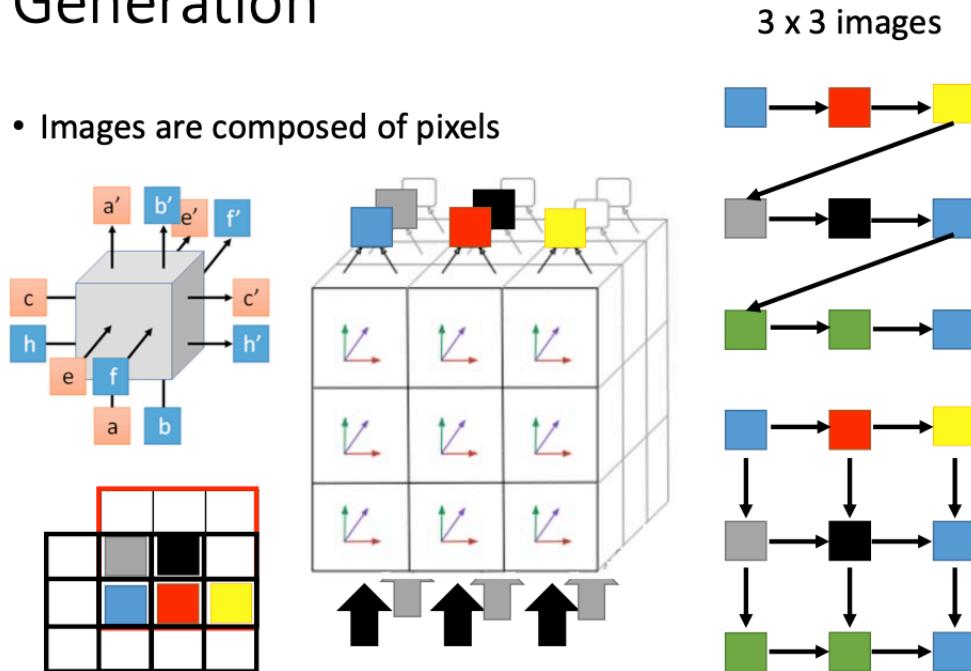


对于一般的generation，如下图的右上部分所示，先根据蓝色pixel生成红色pixel，再根据红色pixel生成黄色pixel，再根据黄色pixel生成灰色pixel，....，并没有考虑pixel之间的位置关系

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

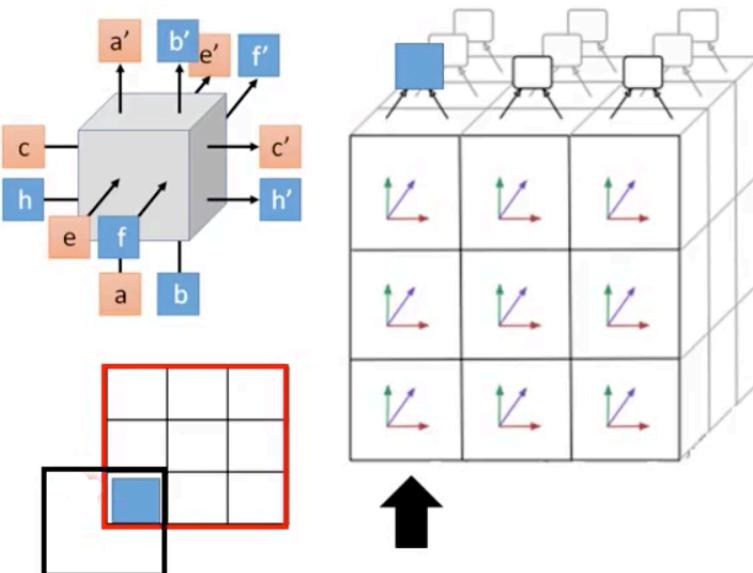
# Generation

- Images are composed of pixels

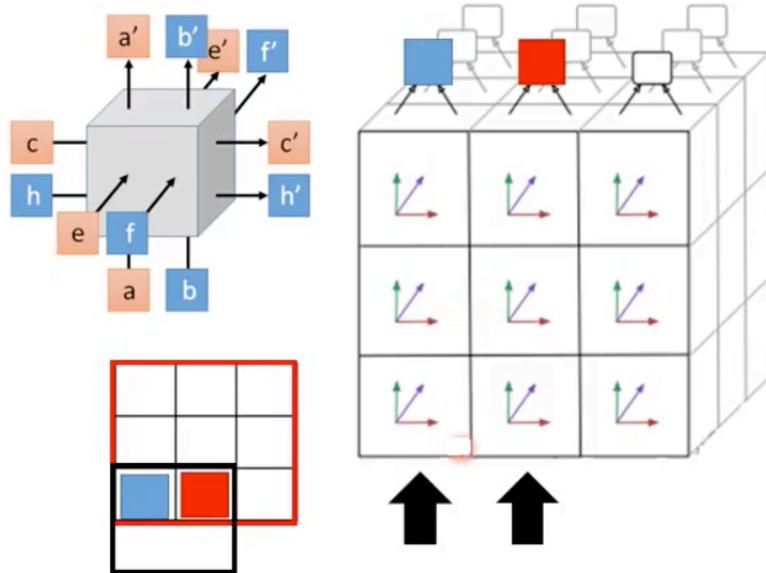


在上图的左上角部分，该lstm块输入了三组参数，也输出了三组参数，把这些方块叠起来就可以达到中部位置图像的效果

对于一个 $3 \times 3$ 的画布，就是我们想要生成的图像的大小，一开始画布上还没有任何内容；现在在画布的左下角放入一个convolution的filter，把其输出放到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



把filter向右移，结合上次蓝色pixel的输出，根据上文的步骤，就可以生成红色的pixel，再把红色pixel放到画布对应的位置



## 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:

### Caption Generation

Given condition:



"A young girl  
is dancing."



### Chat-bot

Given condition:



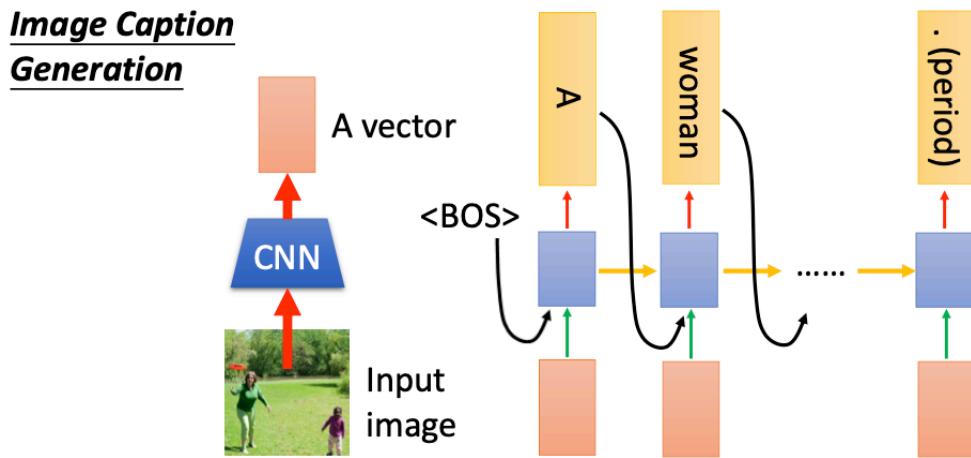
"Hello"

"Hello. Nice  
to see you."



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），先生成了第一个单词"A"，在生成第二个单词时，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

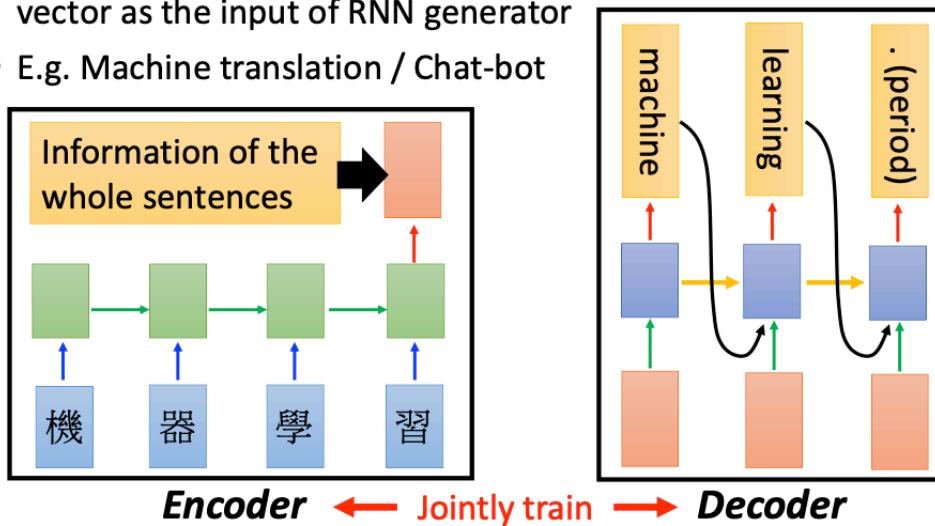
- Represent the input condition as a vector, and consider the vector as the input of RNN generator



RNN也可以用来做机器翻译，比如我们想要翻译“机器学习”，就先把这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）就包含了整个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**  
encoder和decoder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

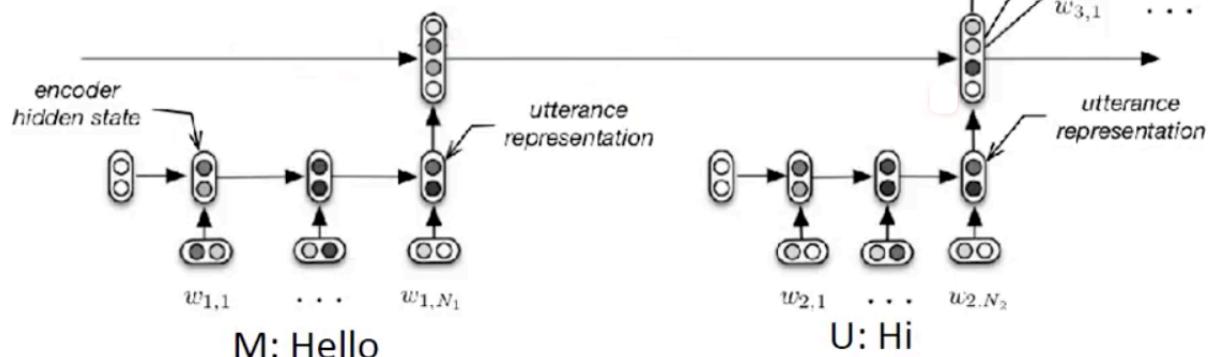
M: Hello

U: Hi

M: Hi

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

Need to consider longer context during chatting

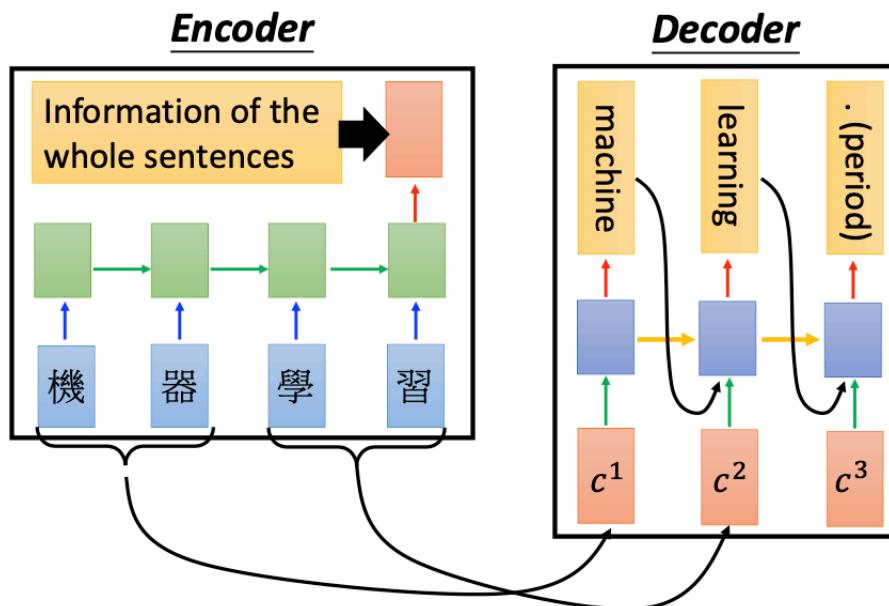


## Attention

### 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



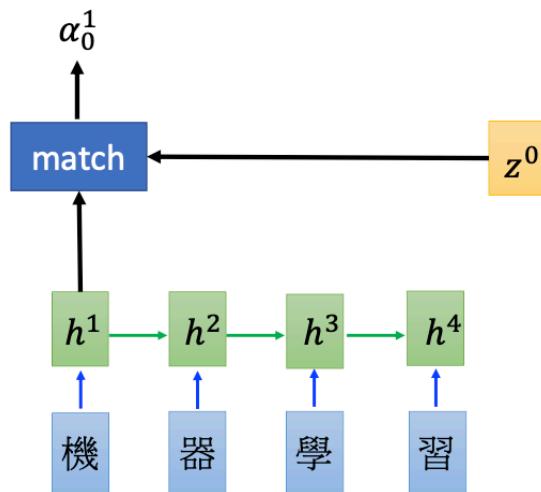
## Machine Translation

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  $\alpha_0^1$

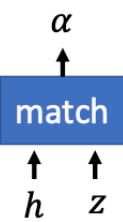
这个match函数可以自己设计，可以有参数，也可以没有参数

# Machine Translation

- Attention-based model



Jointly learned  
with other part  
of the network



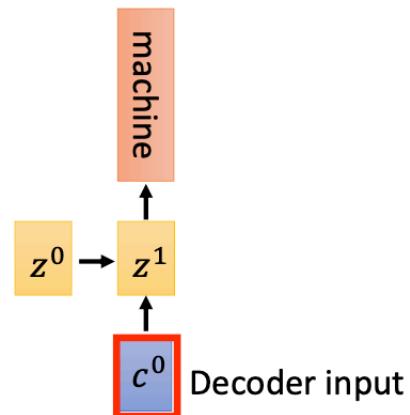
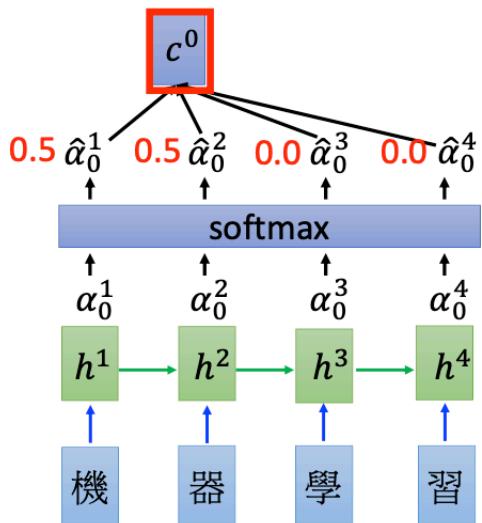
What is **match** ?

Design by yourself

- Cosine similarity of  $z$  and  $h$
- Small NN whose input is  $z$  and  $h$ , output a scalar
- $\alpha = h^T W z$

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

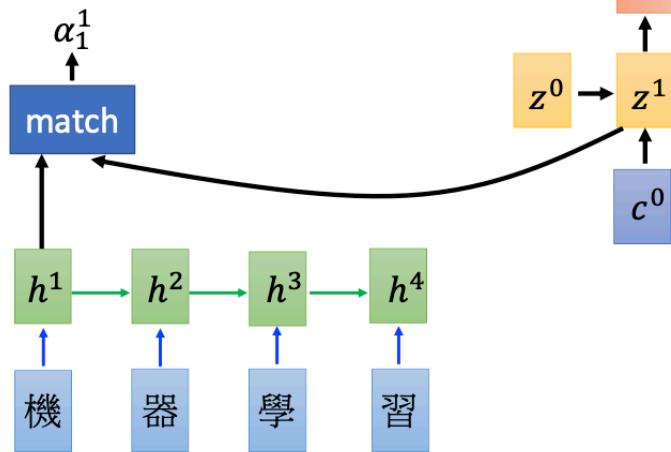
- Attention-based model



$$c^0 = \sum \hat{\alpha}_0^i h^i \\ = 0.5h^1 + 0.5h^2$$

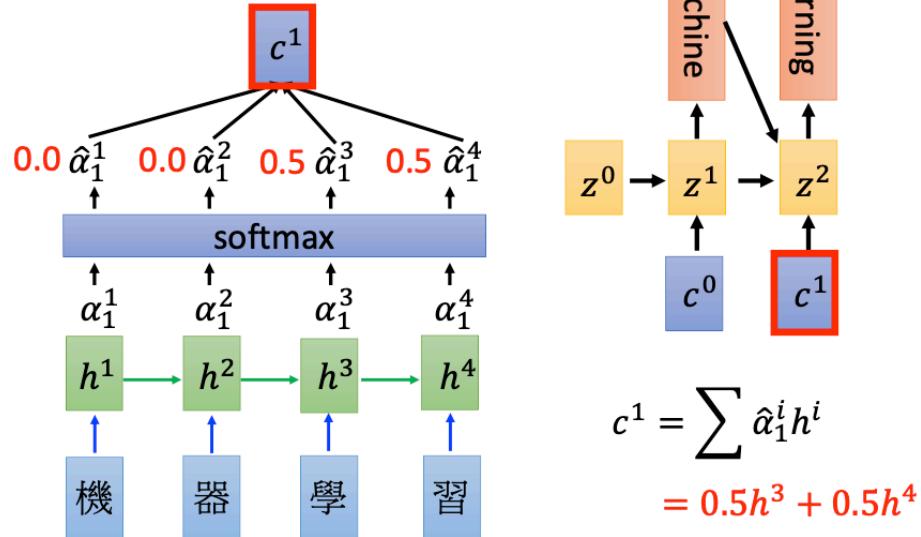
再来计算  $z^1, h^1$  之间的 match 分数  $\alpha_1^1$ ,

- Attention-based model



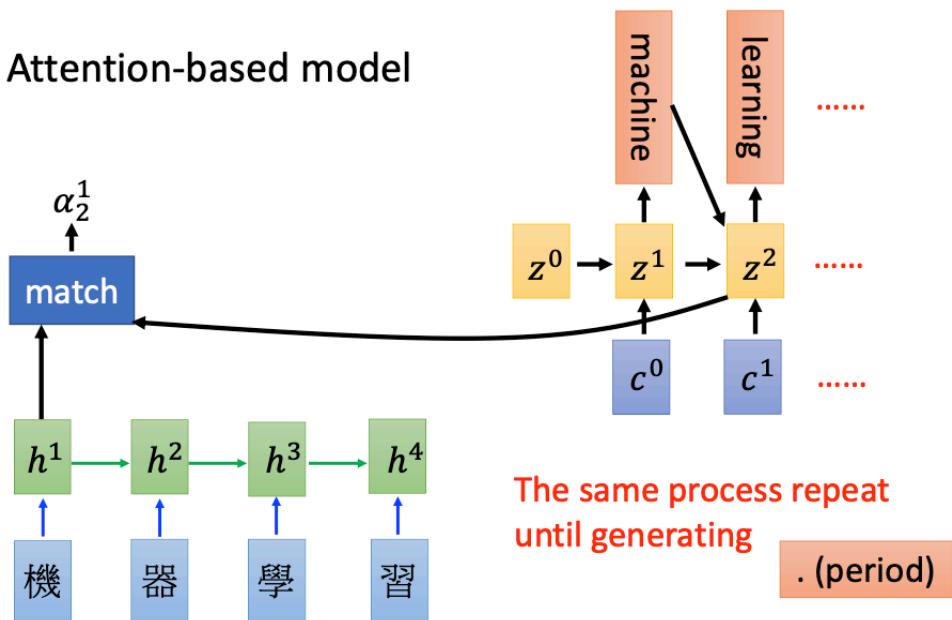
再继续算出 $c^1$ 作为decoder的input, 从而得出第二个单词“learning”

- Attention-based model



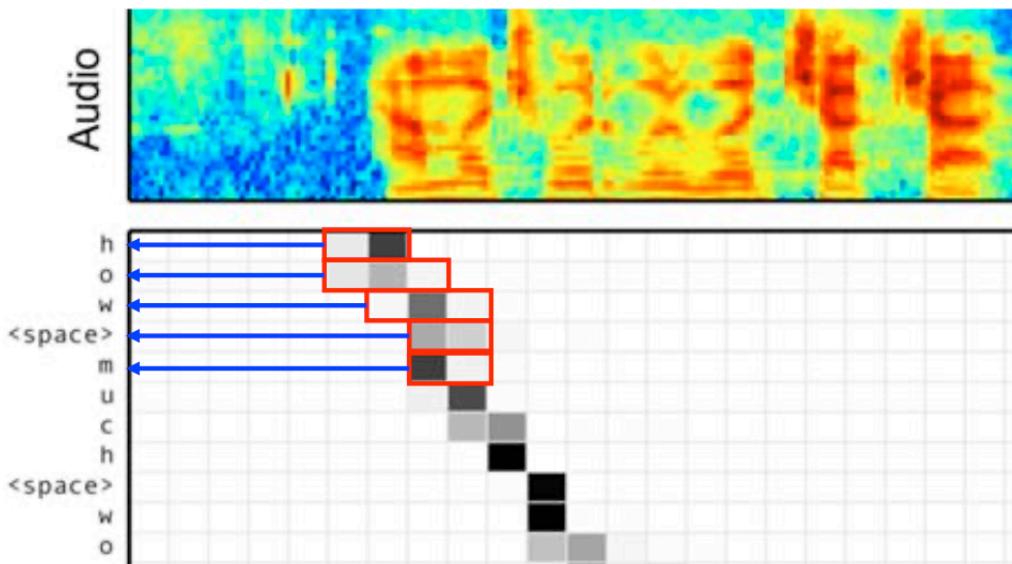
一直重复这个过程, 只到全部翻译完成

- Attention-based model



### 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”

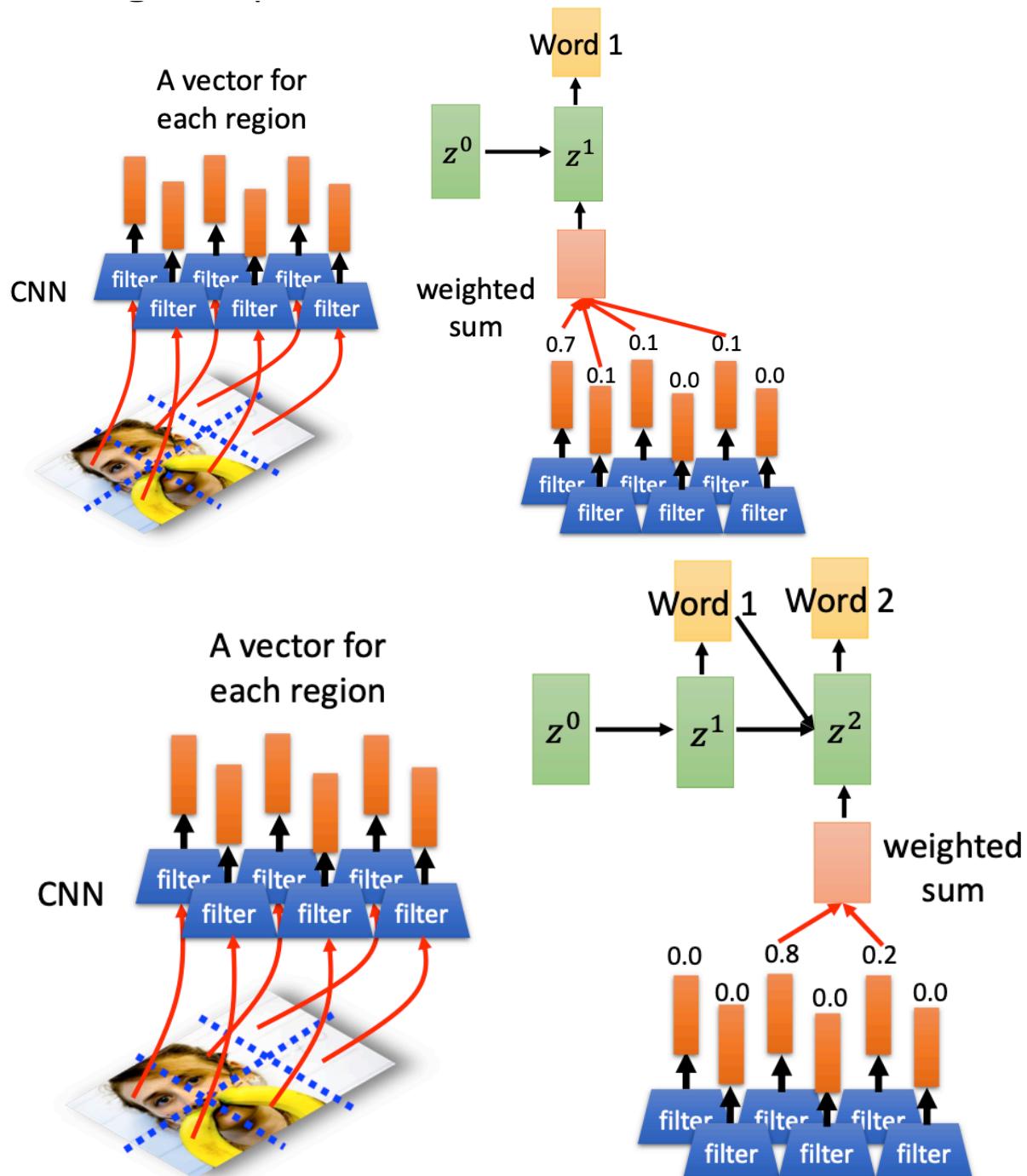


Model	Clean WER	Noisy WER
CLDNN-HMM [22]	8.0	8.9
LAS	14.1	16.5
LAS + LM Rescoring	10.3	12.0

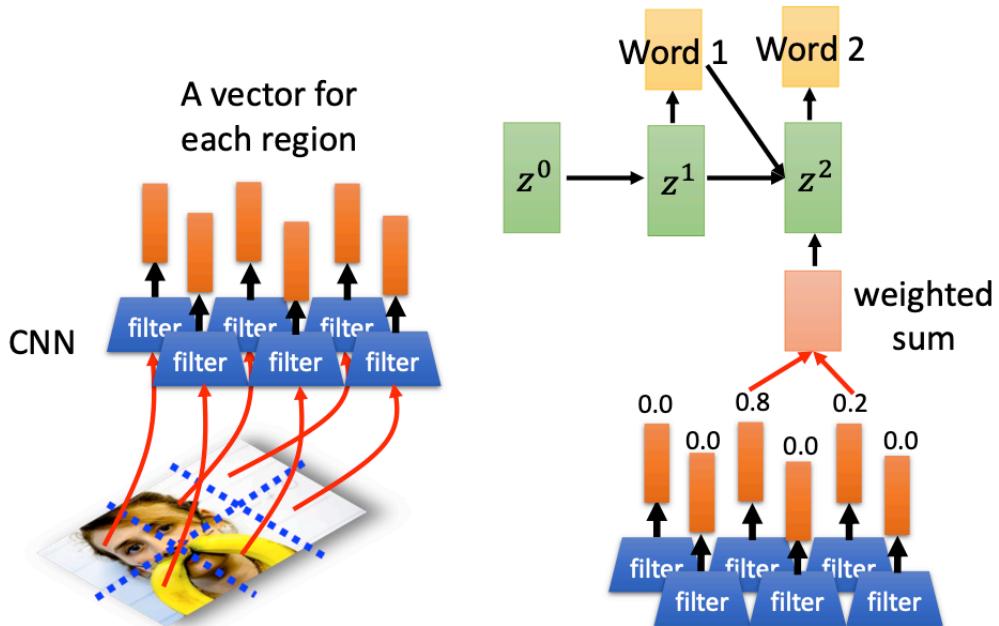
William Chan, Navdeep Jaitly, Quoc V. Le, Oriol Vinyals, “Listen, Attend and Spell”, ICASSP, 2016

### 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Word 2

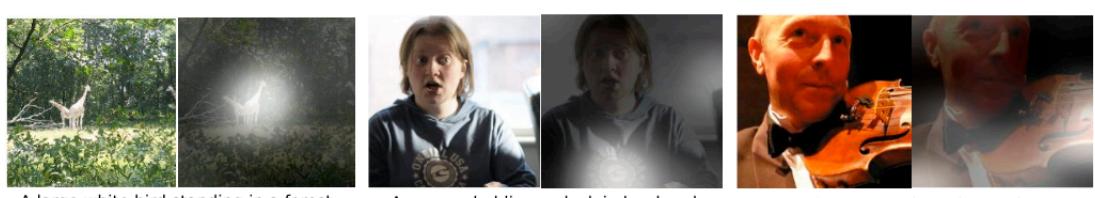


这里是一些具体的例子



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

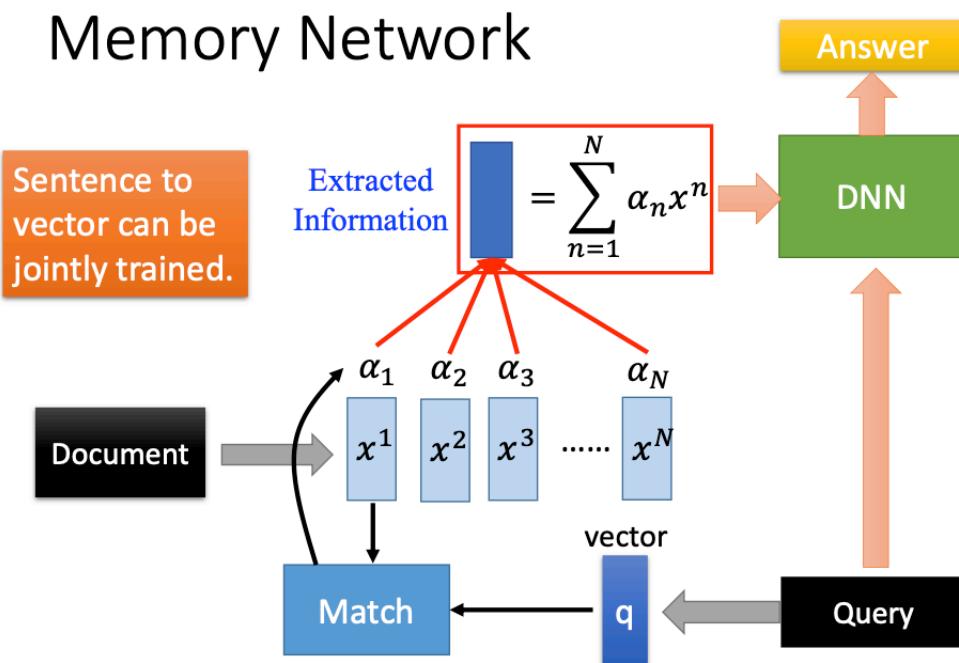
但也会产生一些不好的结果



## Memory Network

memory network是一篇文章，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 $\alpha^1, \dots, \alpha^N$ ，与 $x^i$ 分别相乘，得到extracted information，作为DNN的input，再进行不断的训练，从而得出对应的answer

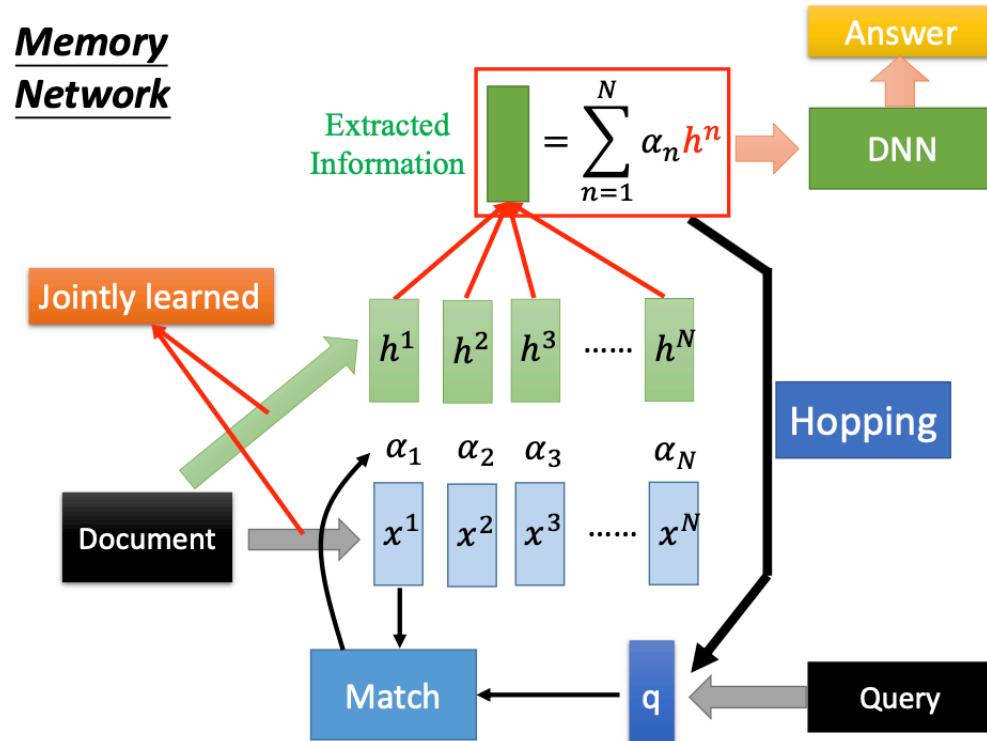


Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus, "End-To-End Memory Networks", NIPS, 2015

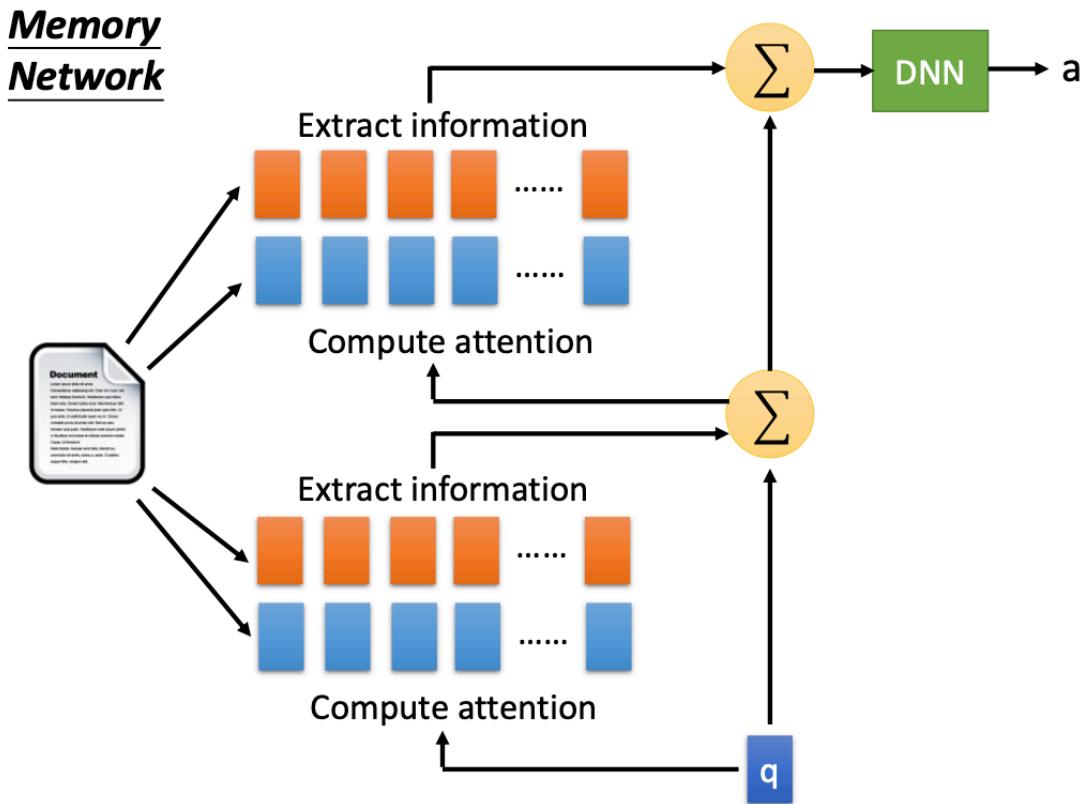
还有一个更加复杂的版本

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



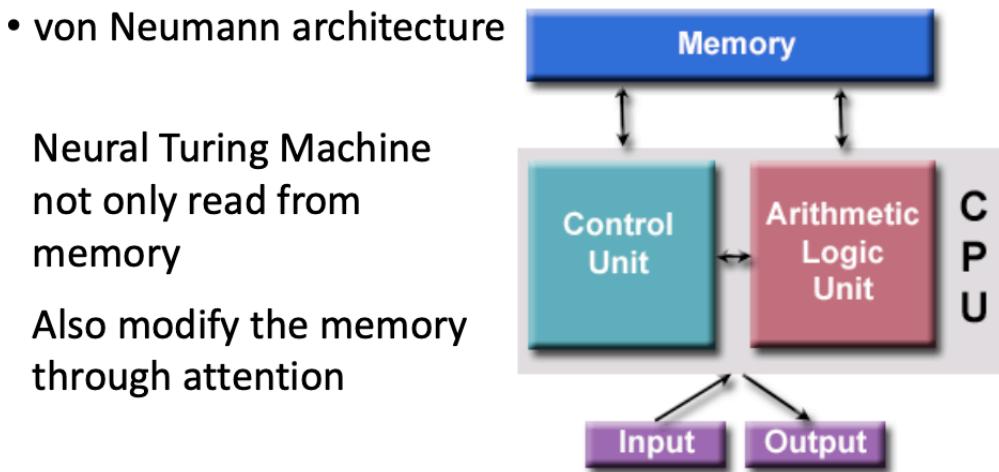
$q$ 可以用来计算attention，计算出extracted information之后，与 $q$ 加起来；上一次sum的结果会再继续参与计算attention，再extract，再进行sum，作为DNN的input，得到最后的答案



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

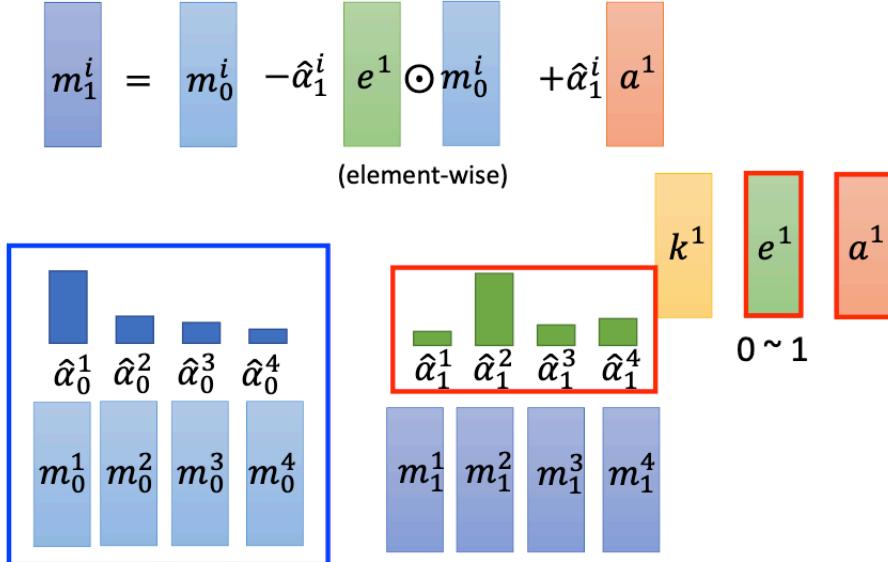
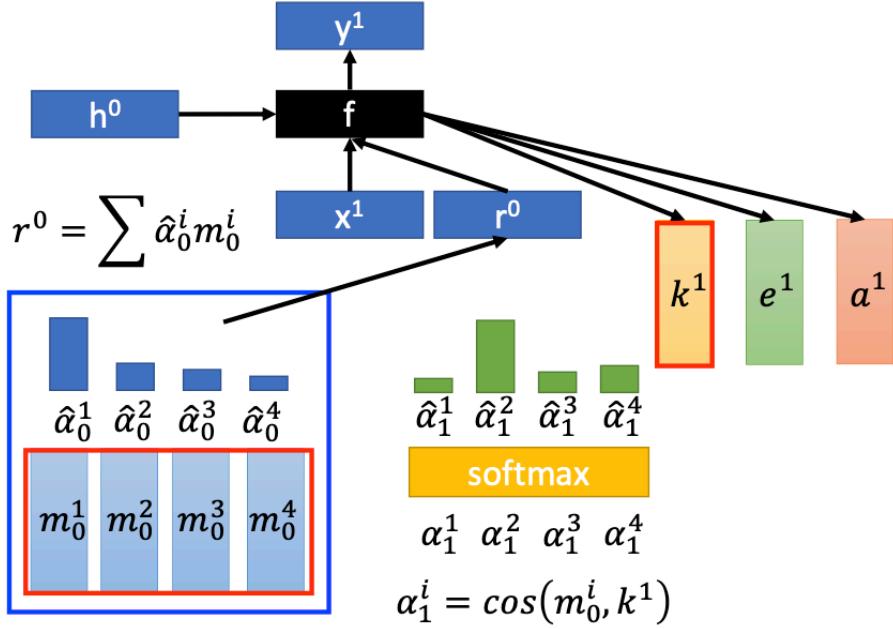
Neural Turing Machine不仅可以读取memory的内容，也可以根据match分数来修改memory的内容；即你不仅可以通过memory读取information，也可以把information写到memory里面去



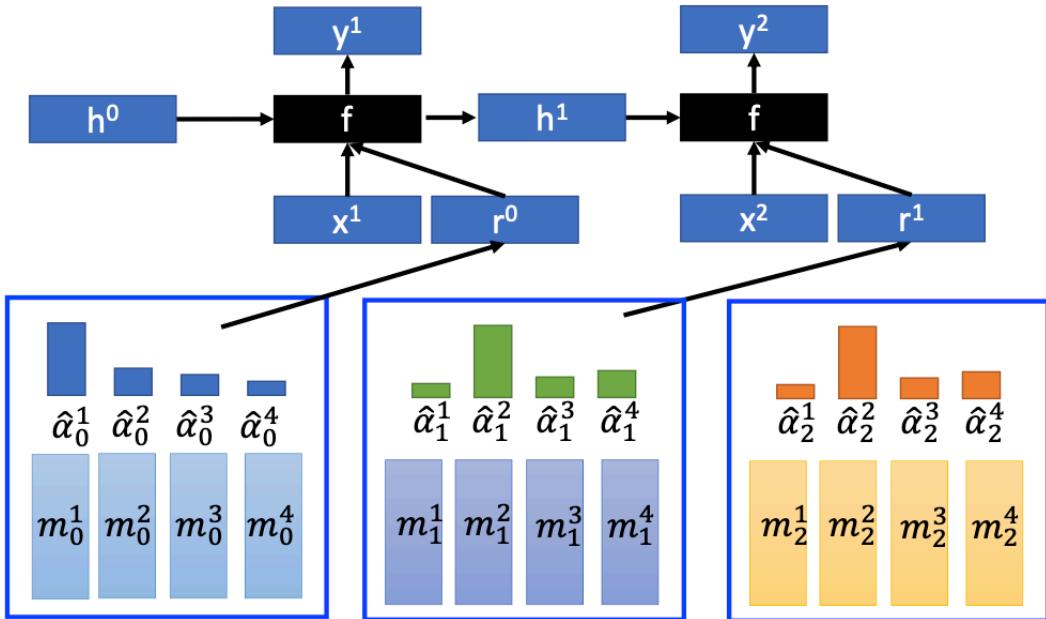
对于初始值的memory  $m_0^i$ ，计算出对应的match分数  $\hat{\alpha}_0^i$ ，在进行weighted sum，得出  $r^0$ ，来作为另外一个network f的input，这个network相当于是一个controller，f可以是DNN、SVM、GRU等；

$r^0, x^1$  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计算加入了  $k^1$ ，即  $\alpha_1^i = \cos(m_0^i, k^1)$ ，再输入softmax，得到新的  $\hat{\alpha}_1^i$  的distribution



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

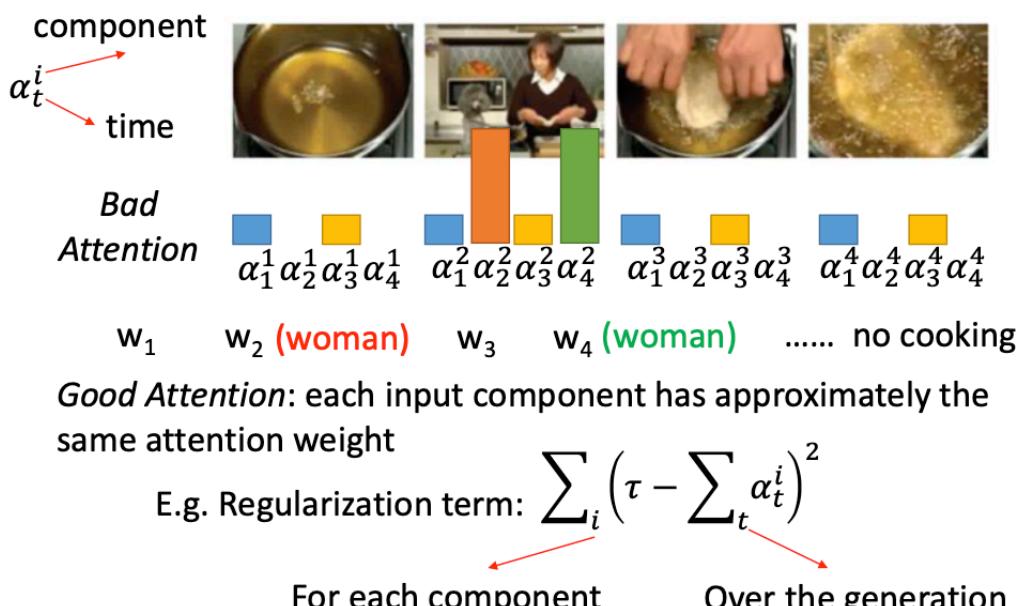


## Tips for Generation

### Attention

$\alpha_t^i$  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 attent 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 attent 一下, 每个 frame 进行 attent 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 $\tau$ ,  $\sum_t \alpha_t^i$ 表示所有attention的sum, 再计算 $\sum_i (\tau - \sum_t \alpha_t^i)^2$ 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

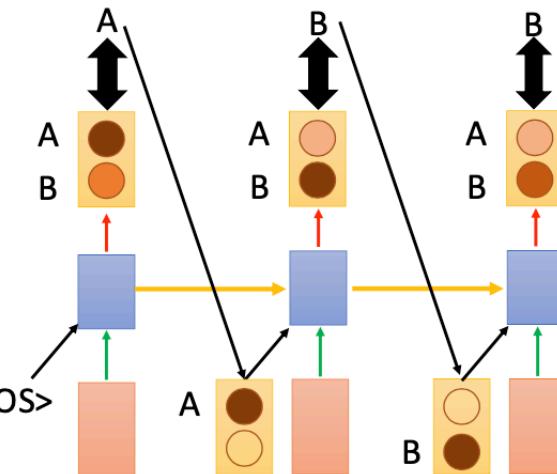
#### • Training

$$C = \sum_t C_t$$

Minimizing cross-entropy of each component

: condition

#### Reference:



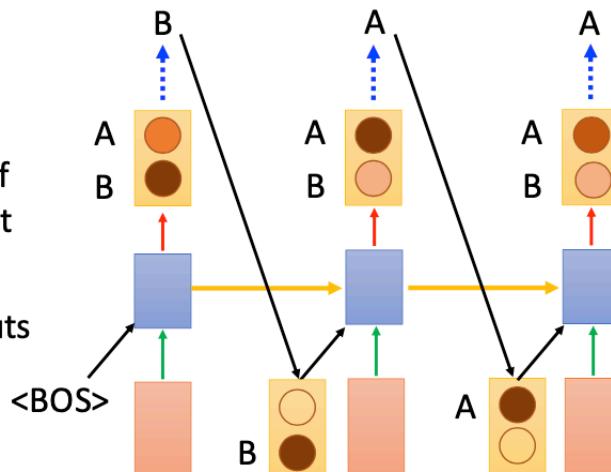
#### • Generation

We do not know the reference

Testing: Output of model is the input of the next step.

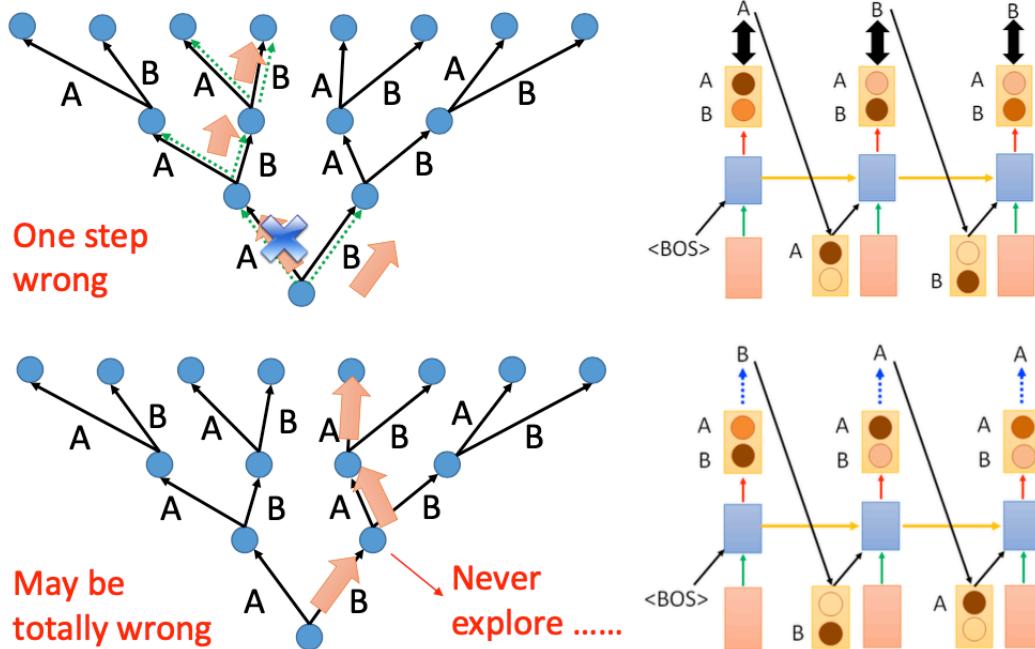
Training: the inputs are reference.

*Exposure Bias*



**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



## 一步錯，步步錯

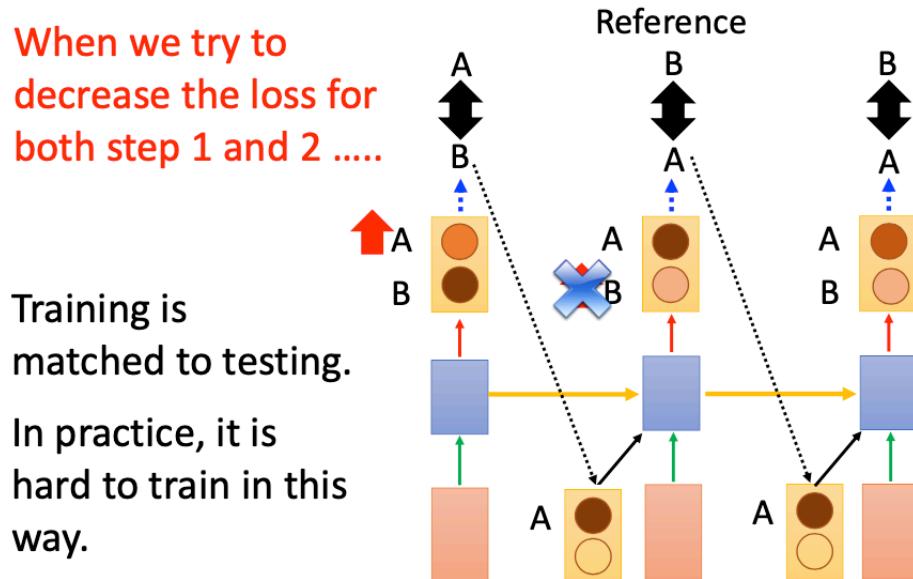
### Modifying Training Process?

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为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

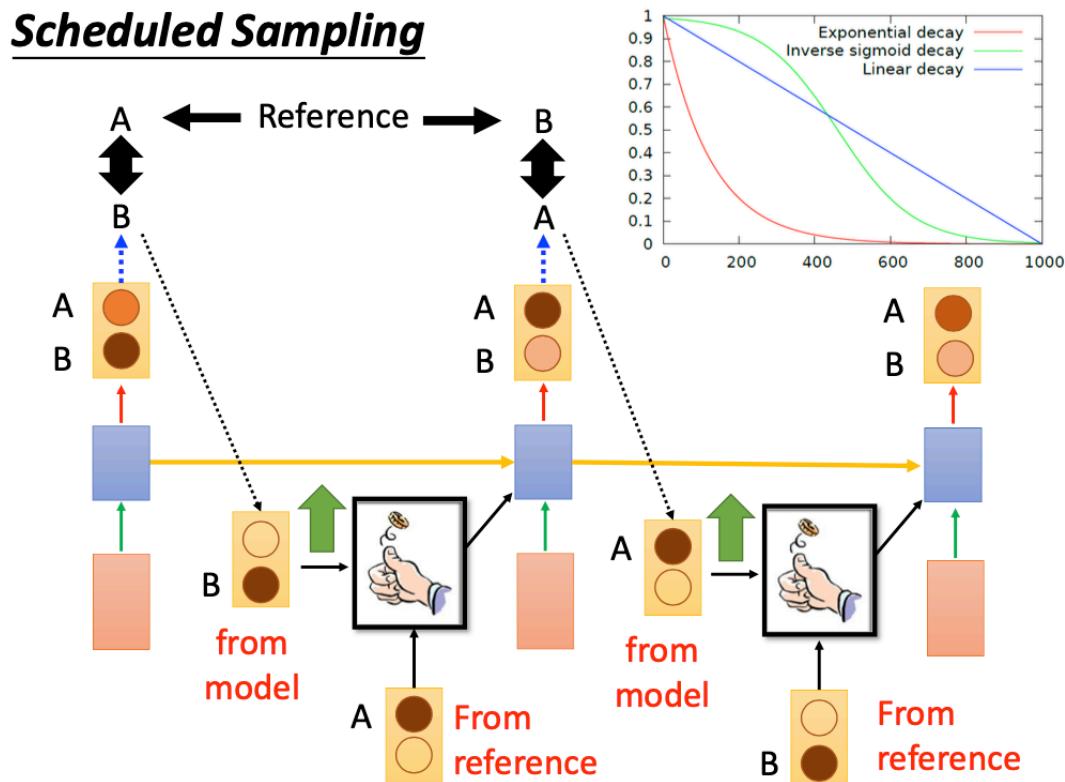


### Scheduled Sampling

由于使用Modifying Training Process很难train，现在我们就使用Scheduled Sampling

对于到底是model里的output，还是reference来作为input，我们可以给一个几率；铜板是正面，就使用model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

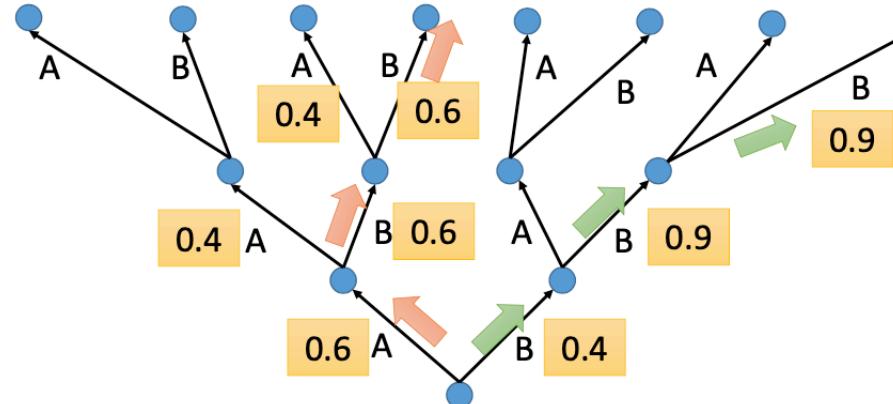
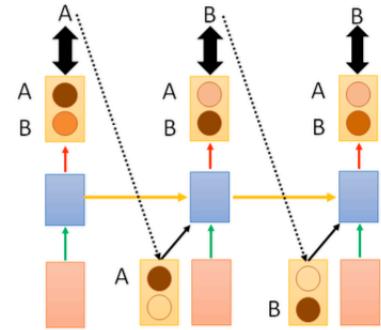
Samy Bengio, Oriol Vinyals, Navdeep Jaitly, Noam Shazeer, Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks, arXiv preprint, 2015

### Beam Search

## Beam Search

The green path has higher score.

Not possible to check all the paths

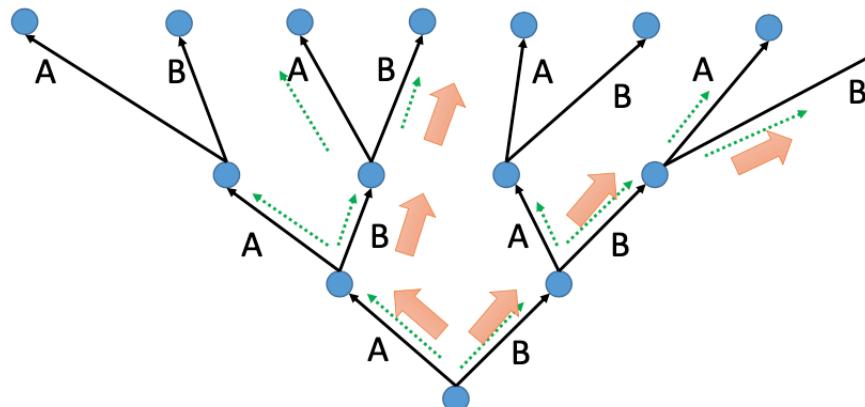
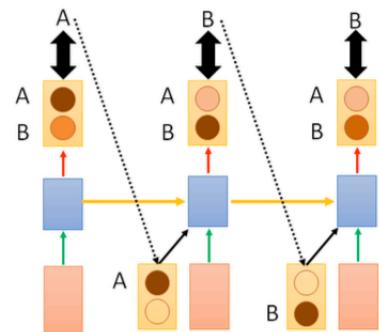


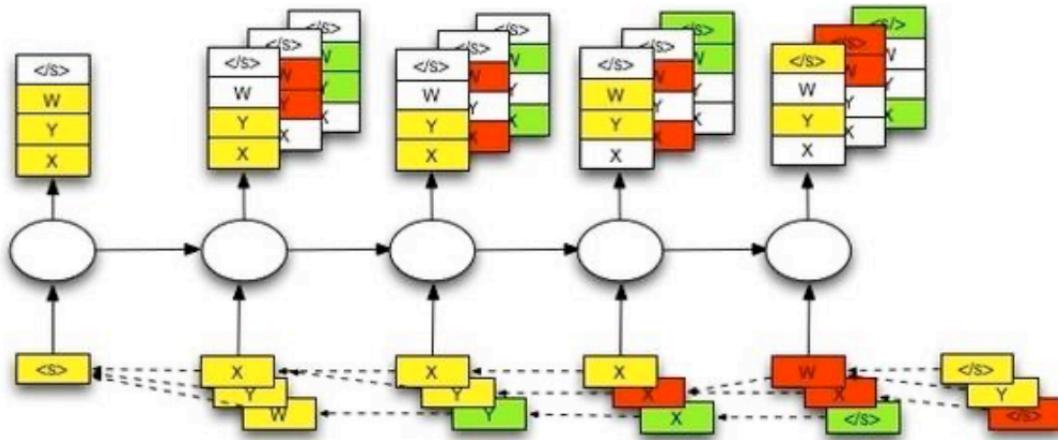
同时走两条path

## Beam Search

Keep several best path at each step

Beam size = 2





The size of beam is 3 in this example.

### Better idea?

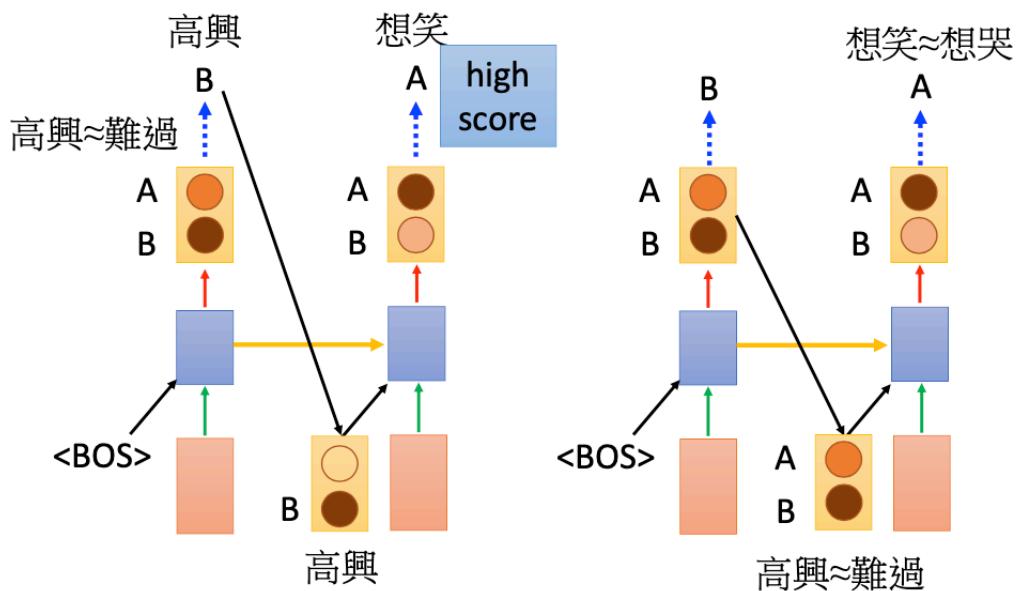
为什么不直接把distribution作为下一个input呢? 即下图中的右图

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

## Better Idea?

U: 你覺得如何?

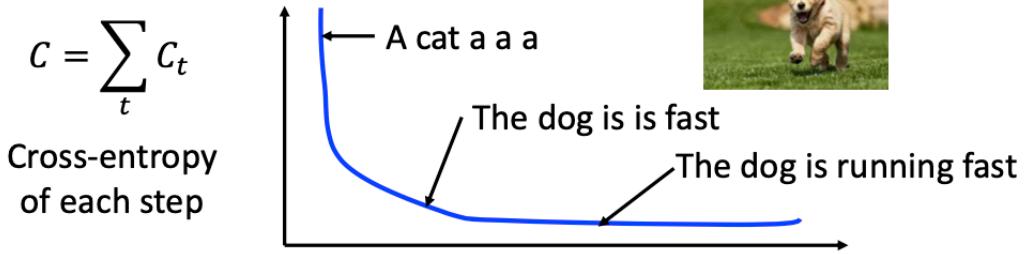
M: 高興想笑 or 難過想哭



Object level v.s. Component level

- Minimizing the error defined on component level is not equivalent to improving the generated objects

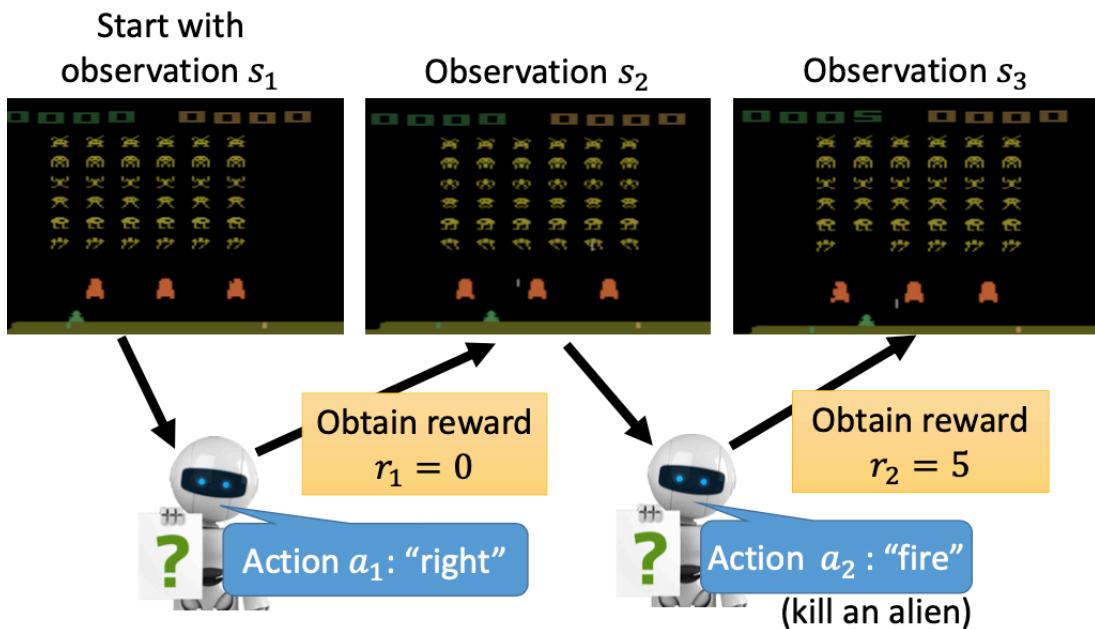
Ref: The dog is running fast



Optimize object-level criterion instead of component-level cross-entropy. object-level criterion:  $R(y, \hat{y})$  Gradient Descent?  
 $y$ : generated utterance,  $\hat{y}$ : ground truth

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

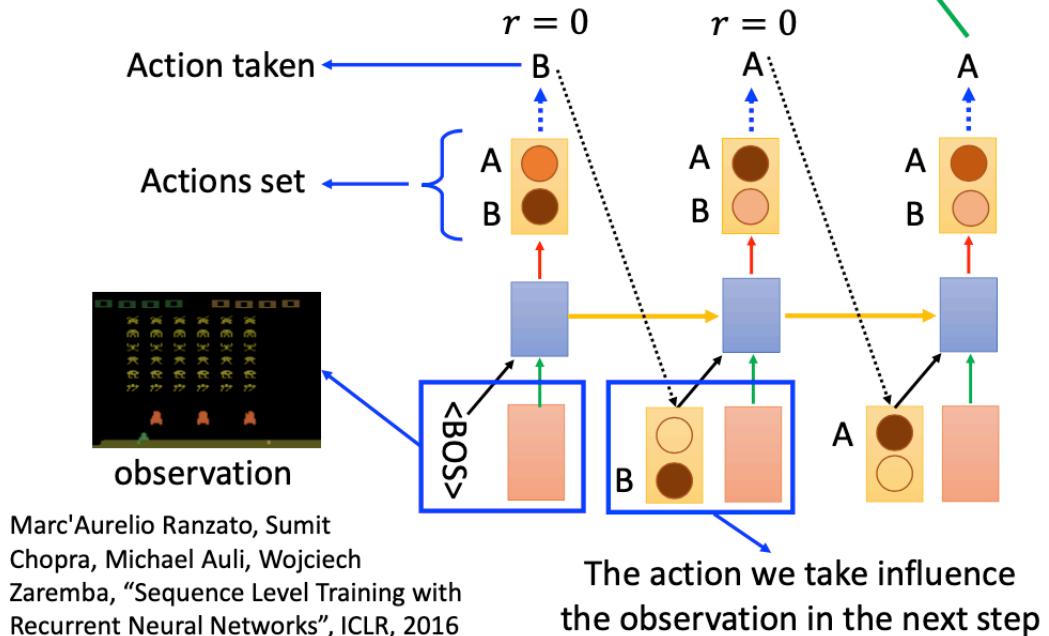
Reinforcement learning?



# Reinforcement learning?

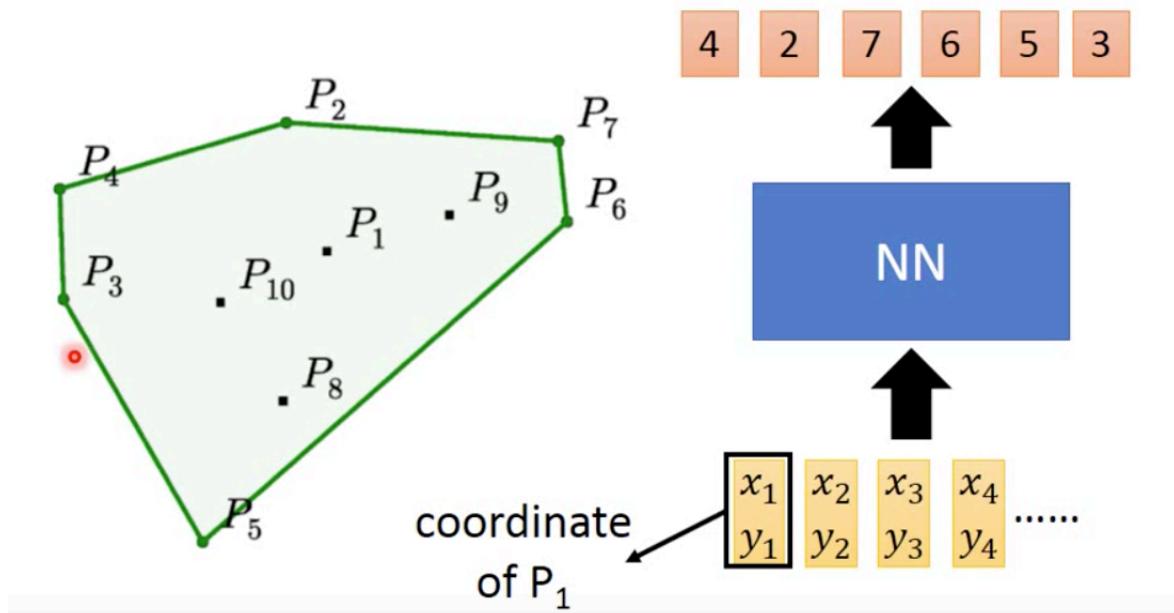
*reward:*

R("BAA", reference)



## Pointer Network

定义如下，找到其中的几个point，将剩下的point包围起来

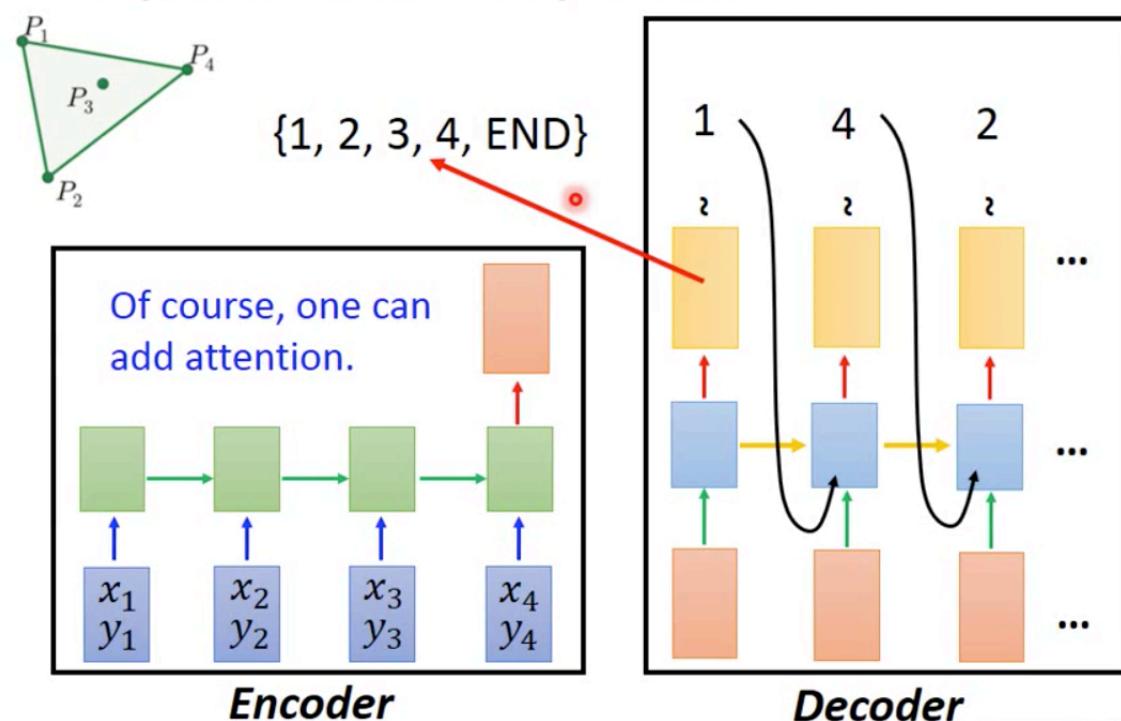


## Seq2seq

input为4个point，但output只有三个point ( $P_1, P_2, P_4$ )，那这个问题还是sequence to sequence吗？

## Problem?

# Sequence-to-sequence?



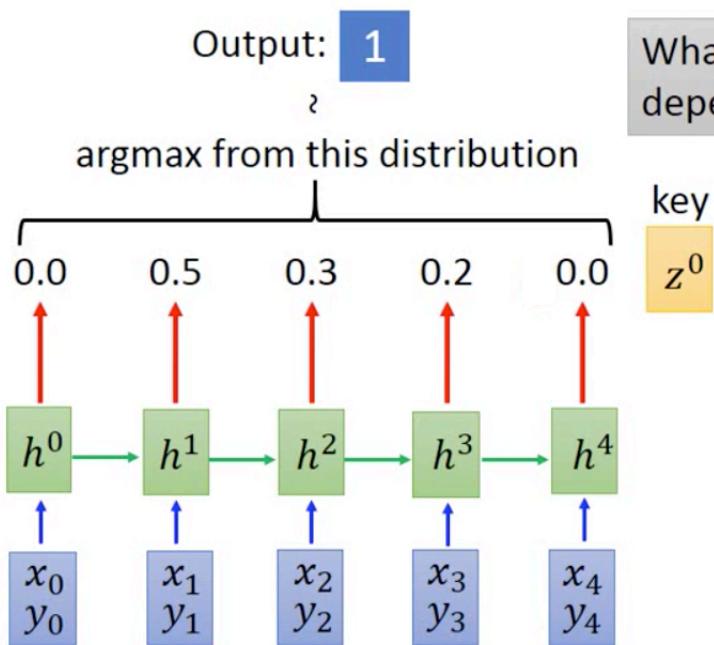
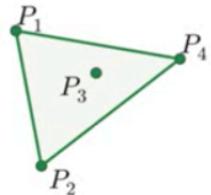
由于output的个数是提前设定好的，并不能动态改变；在training的时候，output很只有50个point，但在testing的时候，output有100个point，由于之前只学习了50个point，因此现在并不能生成51-100之间的点；因此这种做法是不可取的

这里我们对network进行了改进

对于输入的5个point（加入了END），先得到RNN中hidden layer的输出  $(h^0, h^1, h^2, h^3, h^4, h^5)$ ，计算attention weight (match score)，再把这个distribution输入argmax函数，输出概率最大的point，即1

# Pointer Network

$x_0$   
 $y_0$  : END

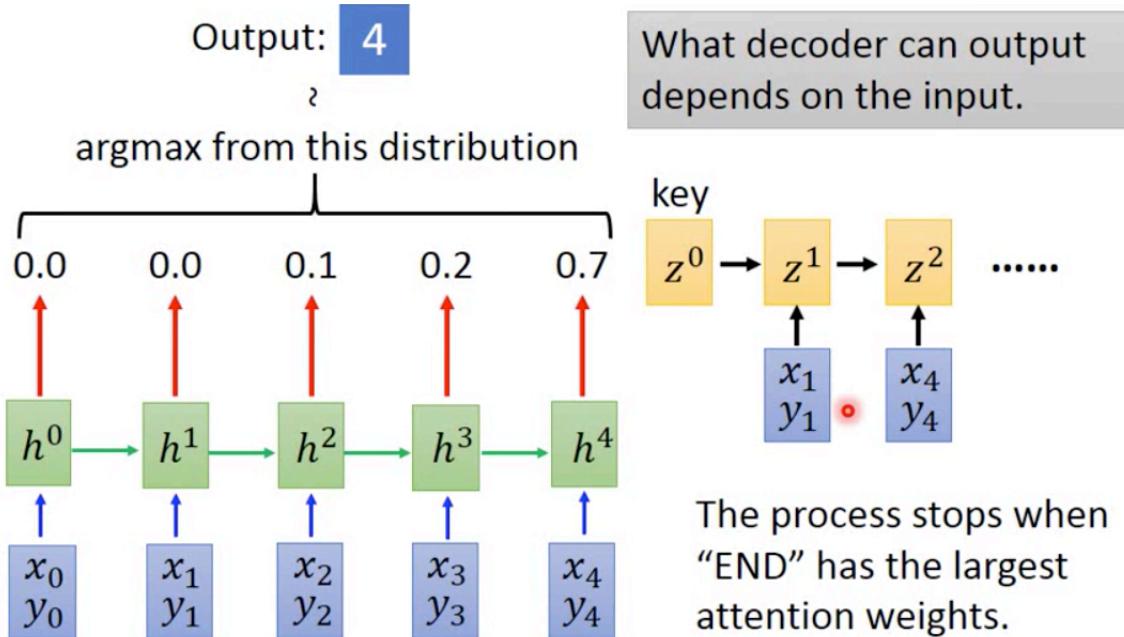


What decoder can output depends on the input.

把这个output 1再输入另外一个RNN network, 其hidden layer的输出为 $z^1$ , 再根据 $z^1, h^i$ 来计算相应的attention weight, 输入argmax函数, 得到output为4

再把point 4作为另外一个RNN network的input, 其hidden layer的输出为 $z^2$ , ....

这个process会一直持续, 知道END出现



What decoder can output depends on the input.

The process stops when "END" has the largest attention weights.

对于改进后的network, 如果现在的input是100个, 那么其output就会有100个不同的选择, 没有了之前的限制

## Applications

对于机器翻译，如果input包含一些地名、人名，并不需要进行翻译，这是我们就可以用pointer network，将这些名词直接copy

### Machine Translation



### Chat-bot

User: X寶你好，我是宏毅

Machine: 宏毅你好，很高興認識你

对于chat-bot，也有一些名词并不需要学习，可以直接copy