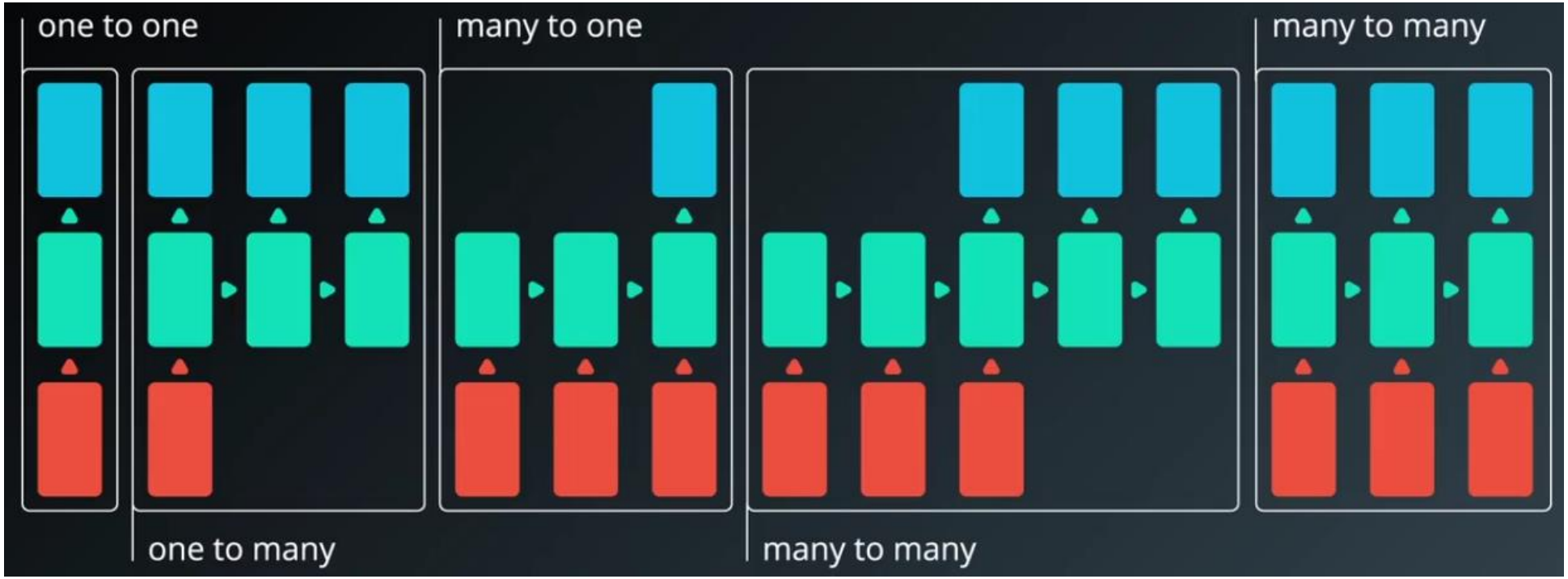
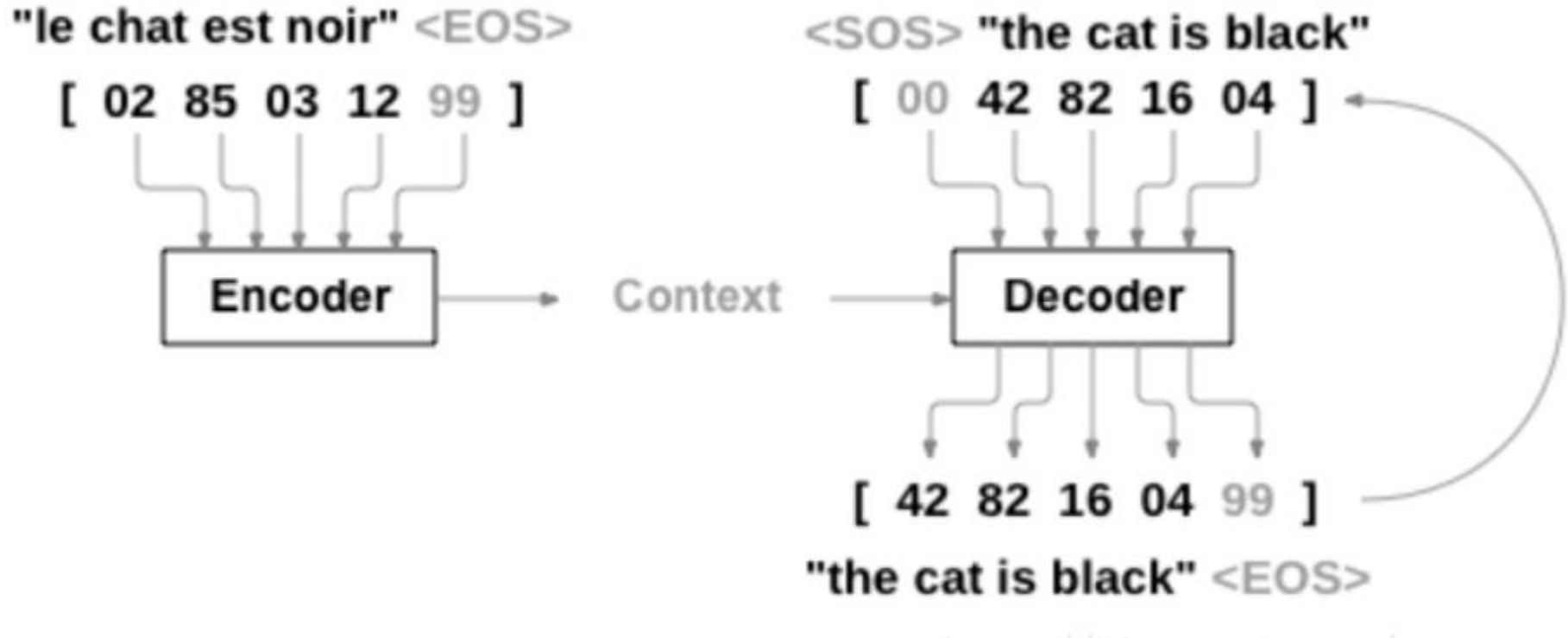
Seq2Seq Attention

Steven Tang

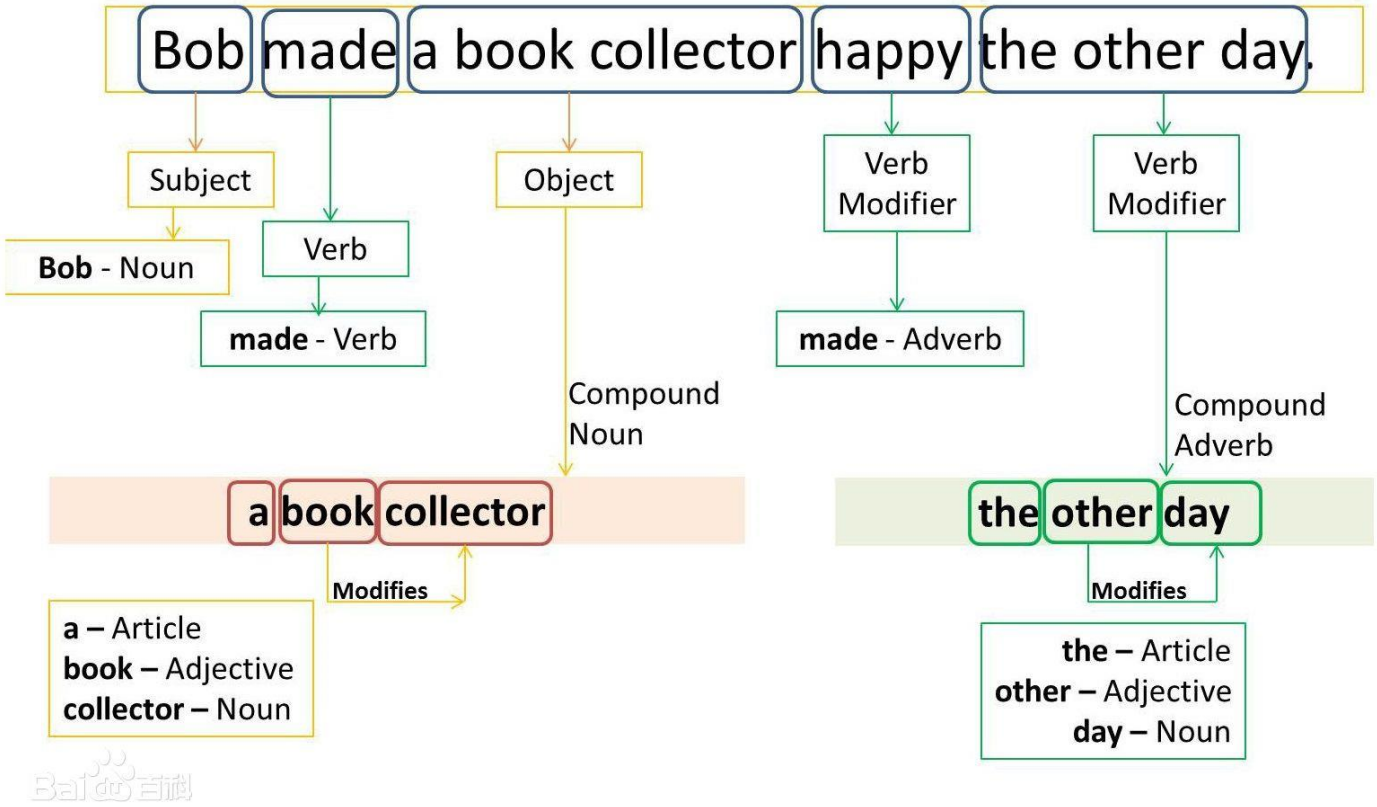
Seq2Seq



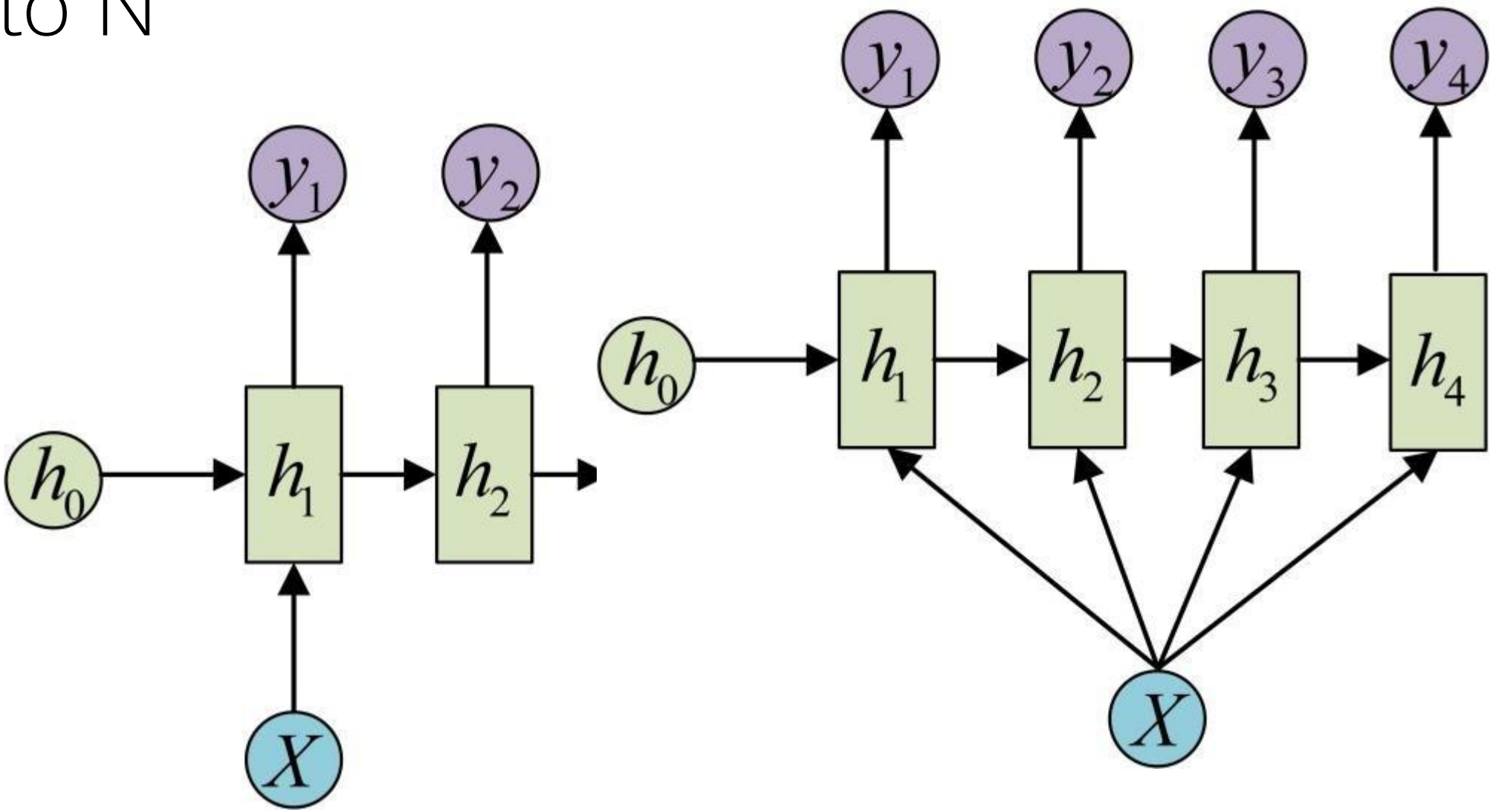
机器翻译


"le chat est noir" <EOS>
[02 85 03 12 99]
Encoder
Context
<SOS> "the cat is black"
[00 42 82 16 04]
Decoder
[42 82 16 04 99]
"the cat is black" <EOS>

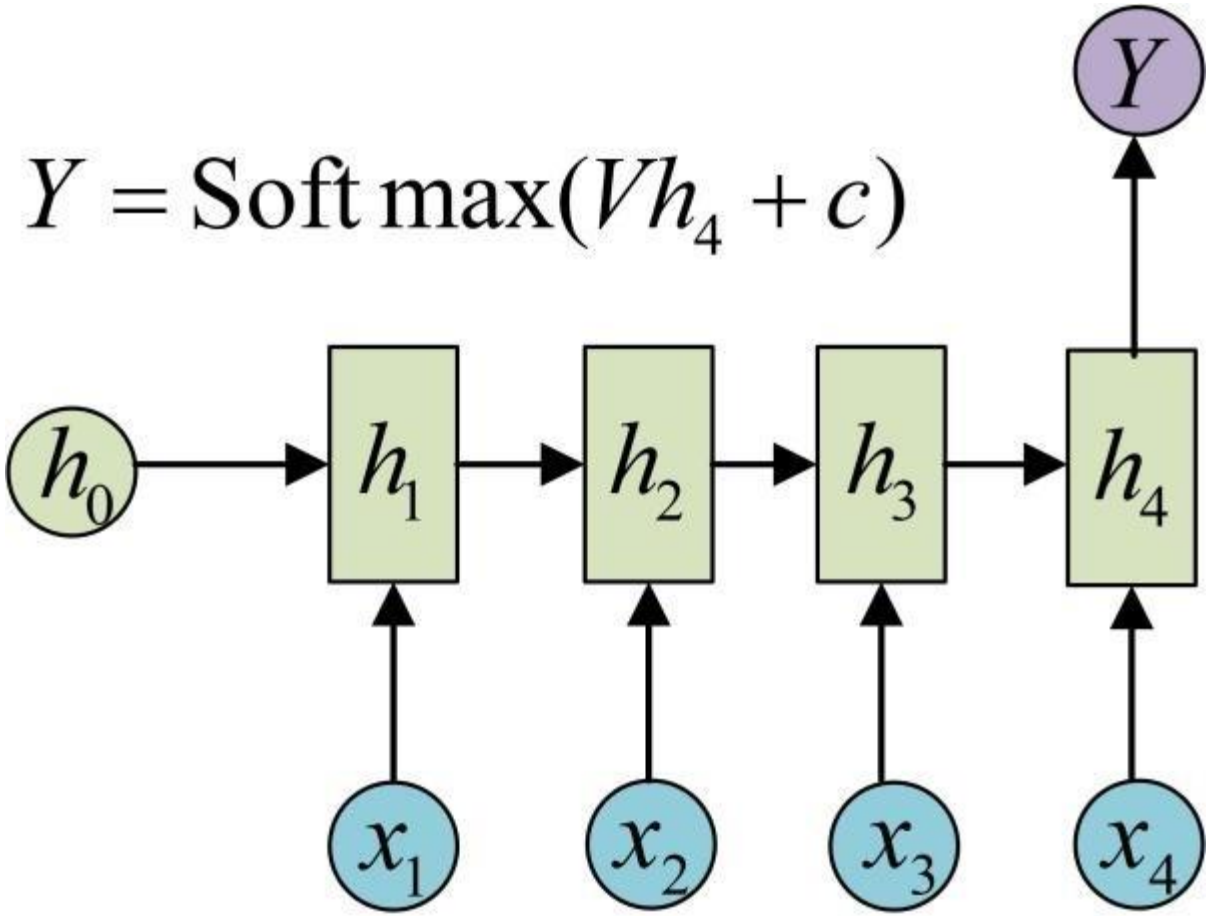
词性标注


Bob made a book collector happy the other day.
Subject
Verb
Object
Verb Modifier
Verb Modifier
Bob - Noun
made - Verb
made - Adverb
Compound Noun
Compound Adverb
a book collector
the other day
Modifies
Modifies
a – Article
book – Adjective
collector – Noun
the – Article
other – Adjective
day – Noun

1 to N



N to 1



Seq2Seq应用(N to M)

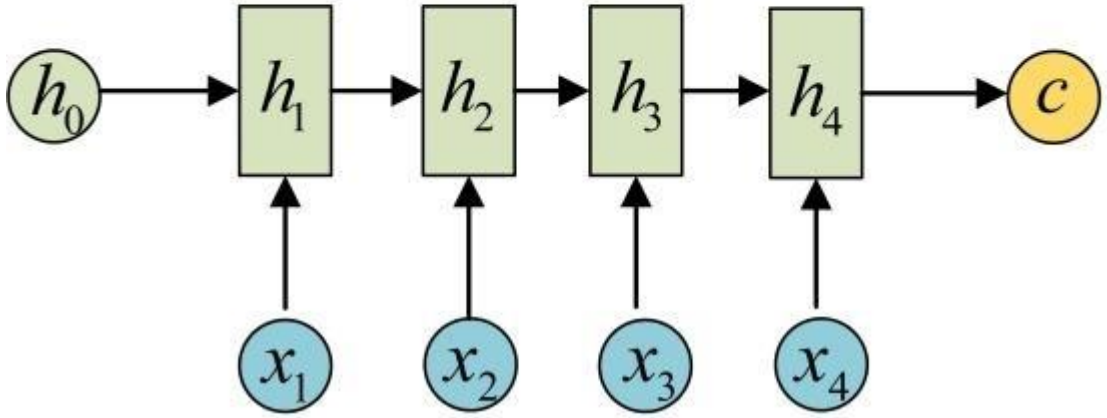
1、机器翻译
2、文章内容摘要
3、问答机器人
4、图片描述

Seq2Seq网络结构

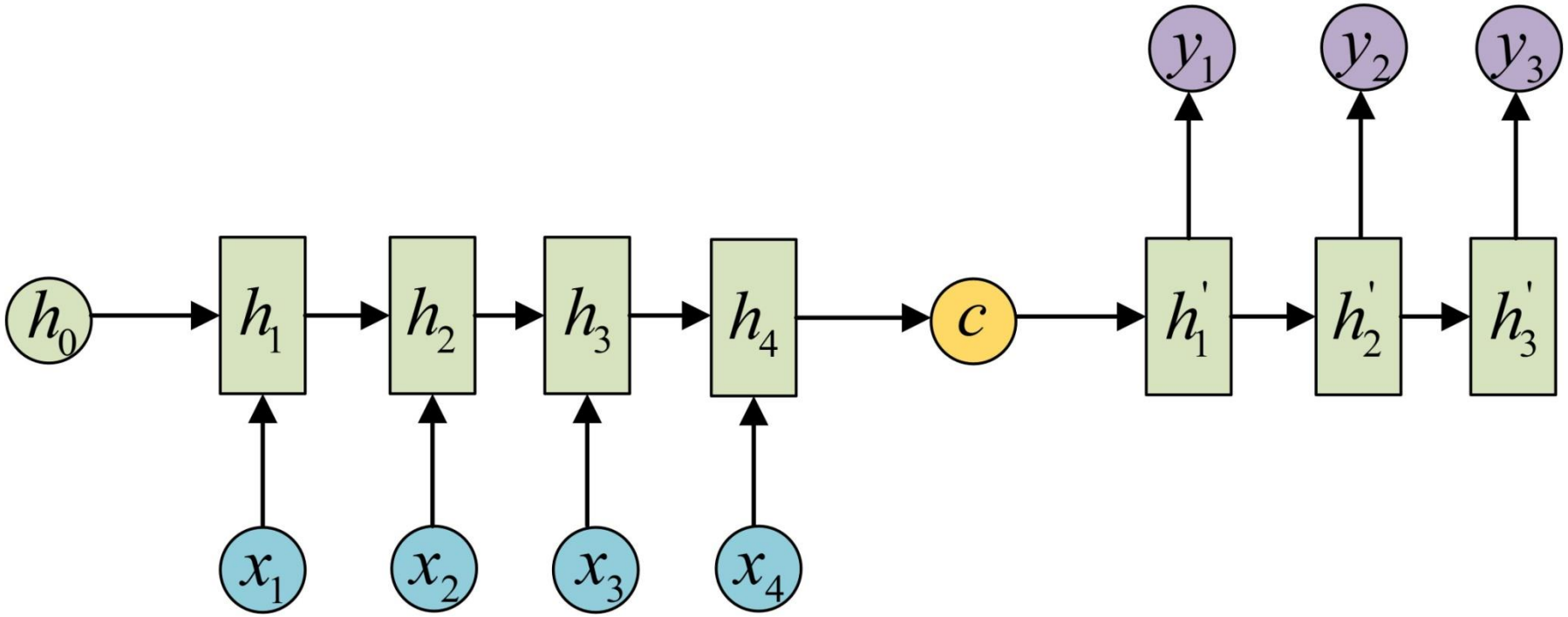
（1）$c = h_4$

（2）$c = q(h_4)$

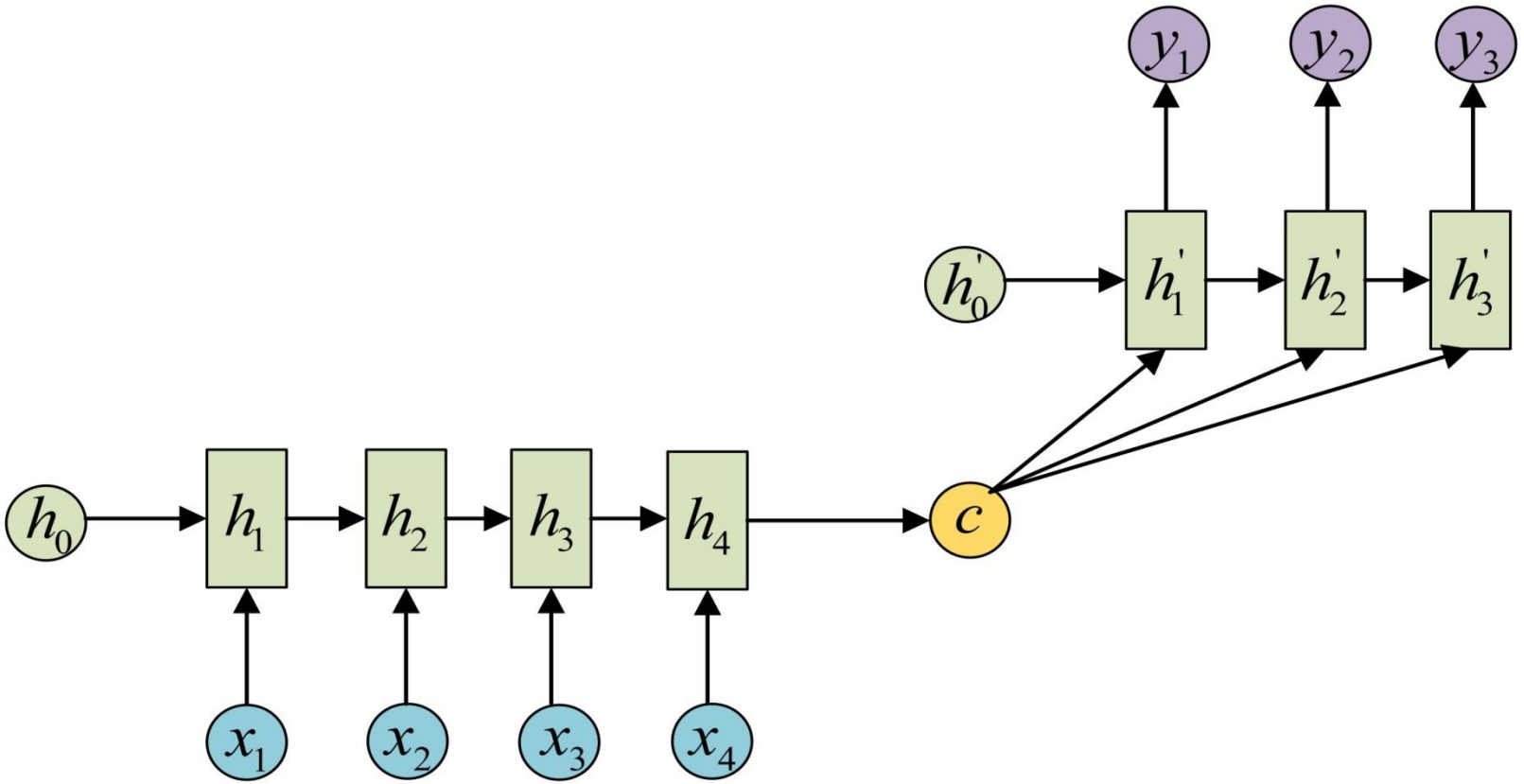
（3）$c = q(h_1, h_2, h_3, h_4)$



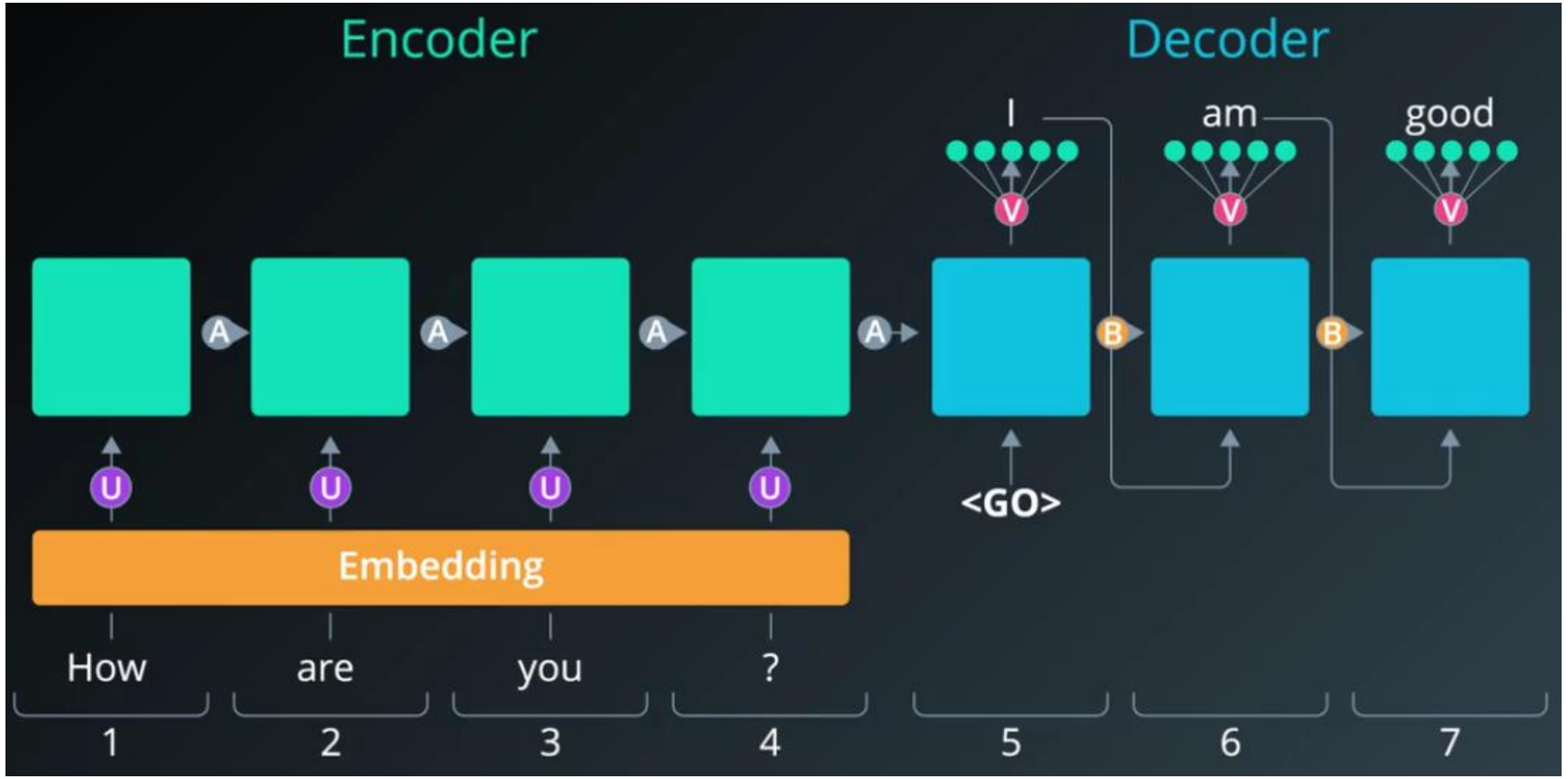
Seq2Seq网络结构



Seq2Seq网络结构



Seq2Seq网络结构



Seq2Seq单词表

0	<PAD>	**空格处填充，**batch中每个句子长度会不同了，此时我们对于短的就直接用 < PAD> 来填充的
1	<EOS>	**表示句子结束**
2	<UNK>	对于一些不常见的词汇，直接用UNK替换掉（例如人名）
3	<GO>	**表示句子开始**
4	How	
5	Are	
6	You	
7	?	
8	I	
9	Am	
10	Good	

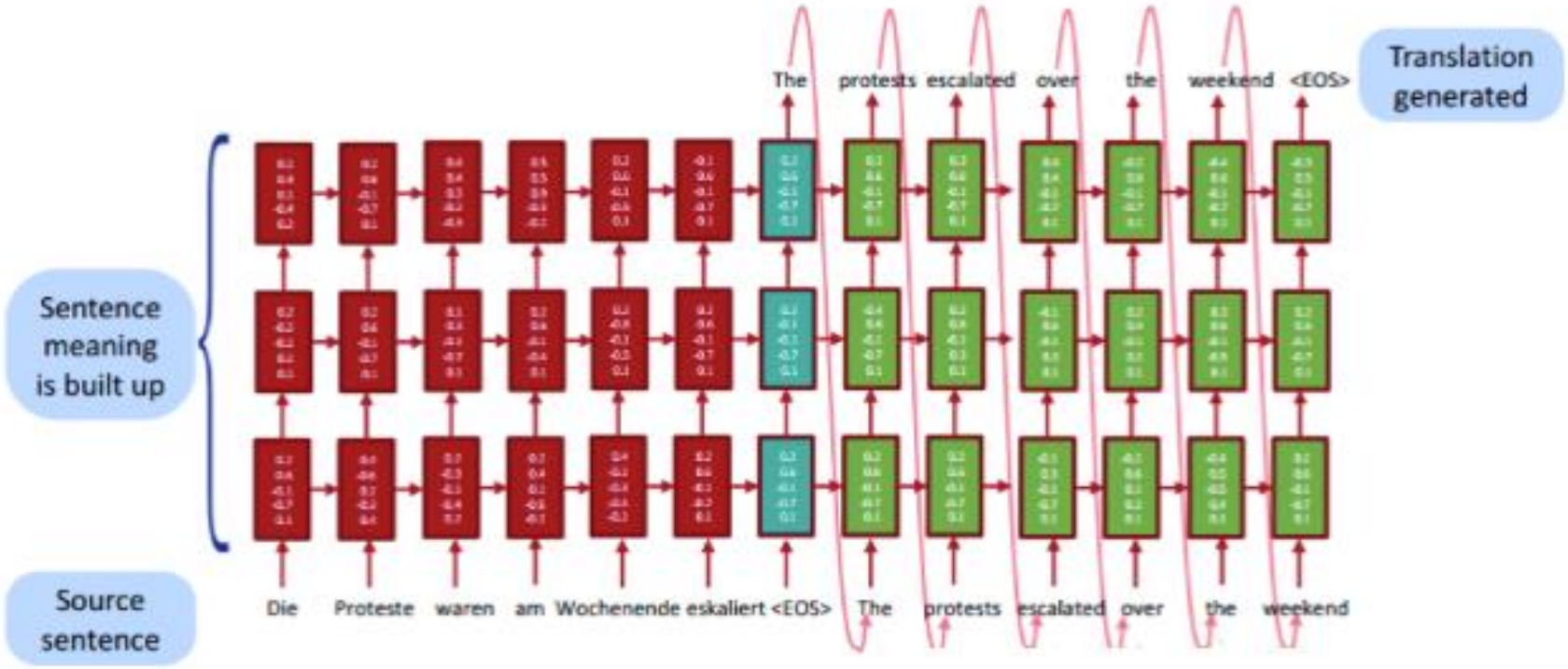
4.5 inputs批处理

Inputs	Target
Can you fly that thing?	Not yet
Is Morpheus alive?	Is Morpheus still alive, Tank?

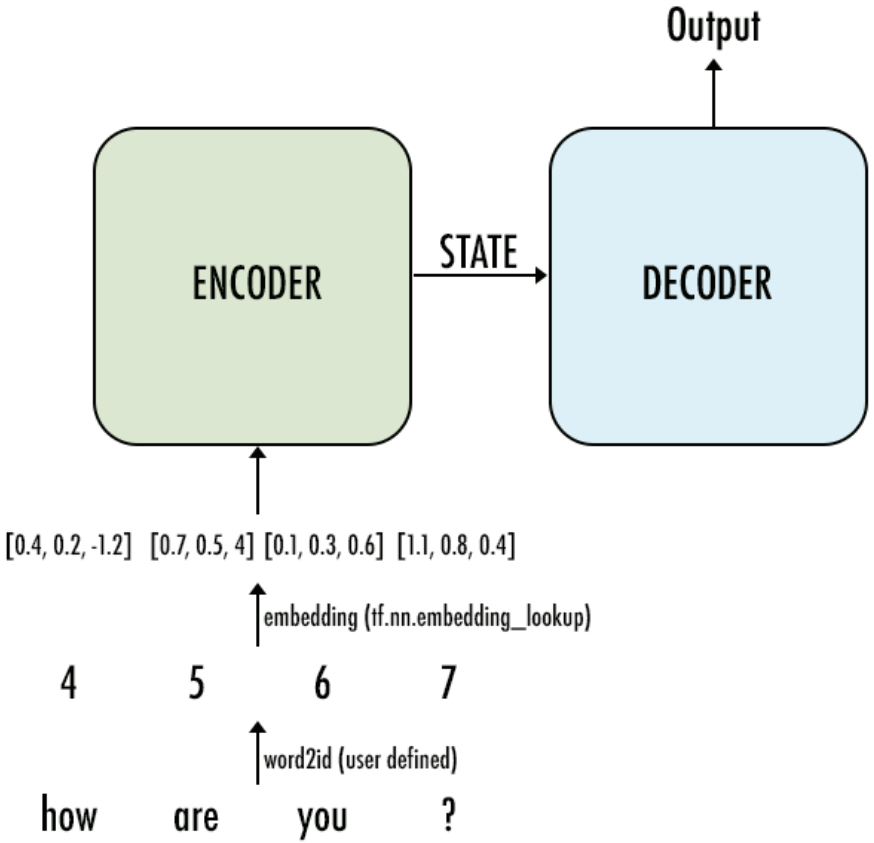
Inputs	Target
can you fly that thing?	not yet
is <UNK> alive?	is <UNK> still alive, <UNK>?

can	**you**	**fly**	**that**	**thing**	**?**
Is	<UNK>	Alive	?	<PAD>	<PAD>
4	**5**	**6**	**7**	**8**	**9**
10	2	11	9	0	0

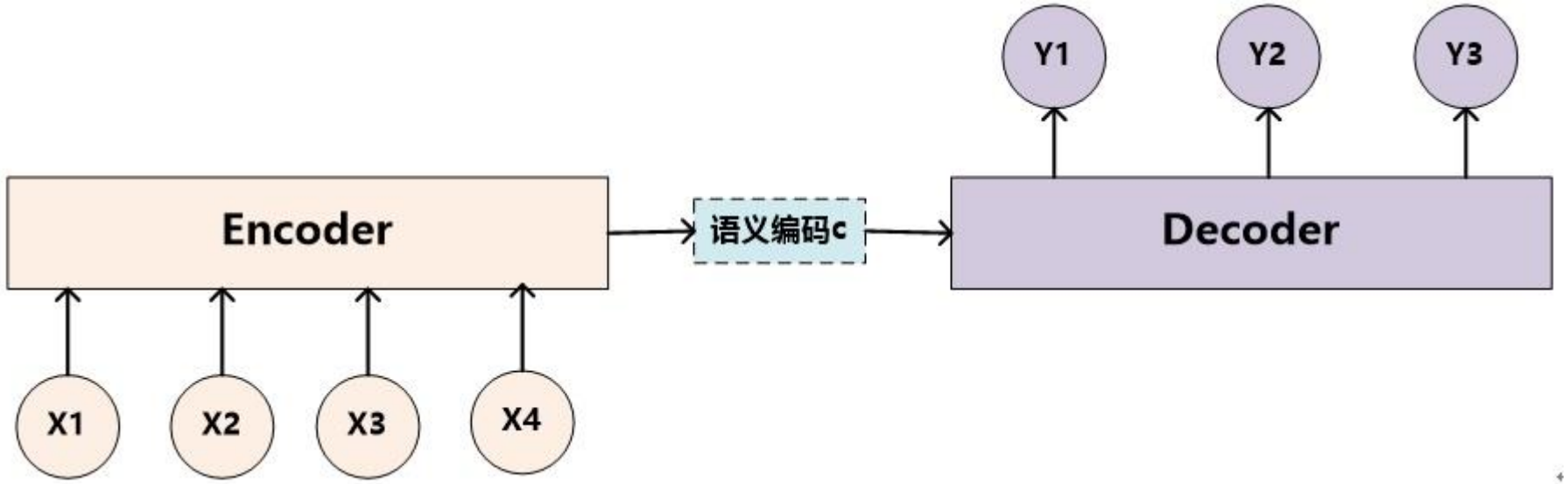
Seq2Seq网络结构



架构范式


Output
ENCODER
STATE
DECODER
[0.4, 0.2, -1.2] [0.7, 0.5, 4] [0.1, 0.3, 0.6] [1.1, 0.8, 0.4]
embedding (tf.nn.embedding_lookup)
4 5 6 7
word2id (user defined)
how are you ?

Encoder-decoder 结构



$$\mathbf{Source} = \langle x_1, x_2 \dots x_m \rangle$$

$$\text{Target} = \langle y_1, y_2 \dots y_n \rangle$$

$$C = \mathcal{F}(x_1, x_2 \dots x_m)$$

$$y_i = \mathcal{G}(C, y_1, y_2 \dots y_{i-1})$$

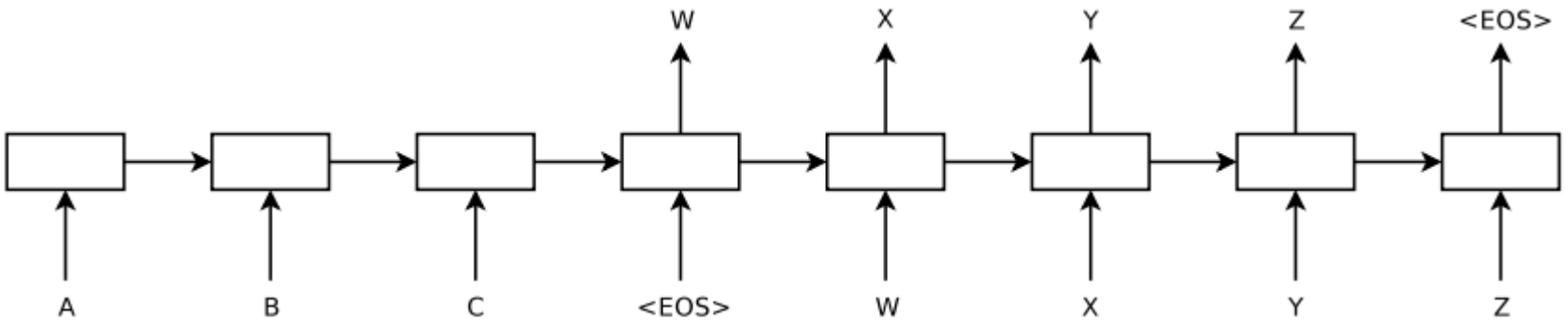
Encoder-decoder 结构

- 1、该模型有两个输入：一个是x输入作为Encoder的输入，另一个是y输入作为Decoder的输入，所以在seq2seq的训练中，标签y既参与计算loss，又参与节点运算。

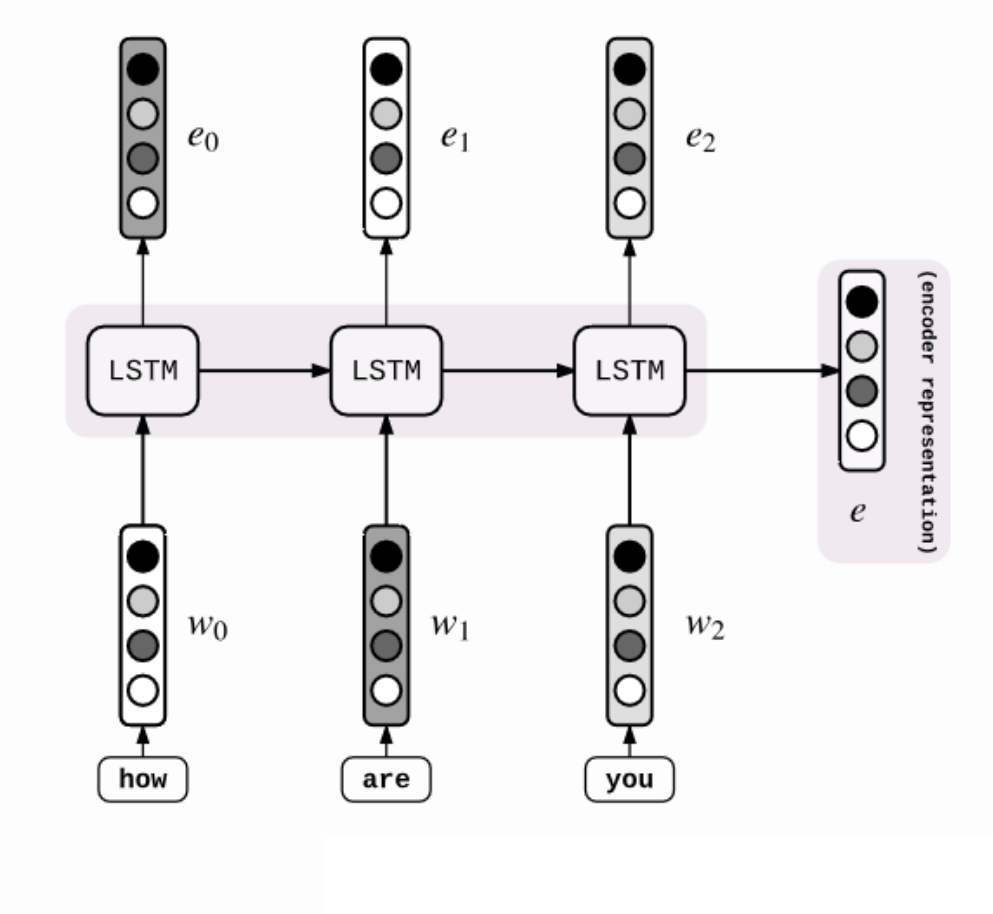
- 2、C 节点是Encoder最后一个时间步长的隐藏状态，它将作为Decoder中cell的初始状态。

- 3、在训练中，Decoder的隐藏状态h_t就由上一个隐藏状态h_(t-1)，y_(t-1)和C三部分构成。
- 为什么？直接预测一条序列有难度，有点像分开预测单个词，给输入序列的信息，以及真实的上一个词的信息，预测下一个词的信息

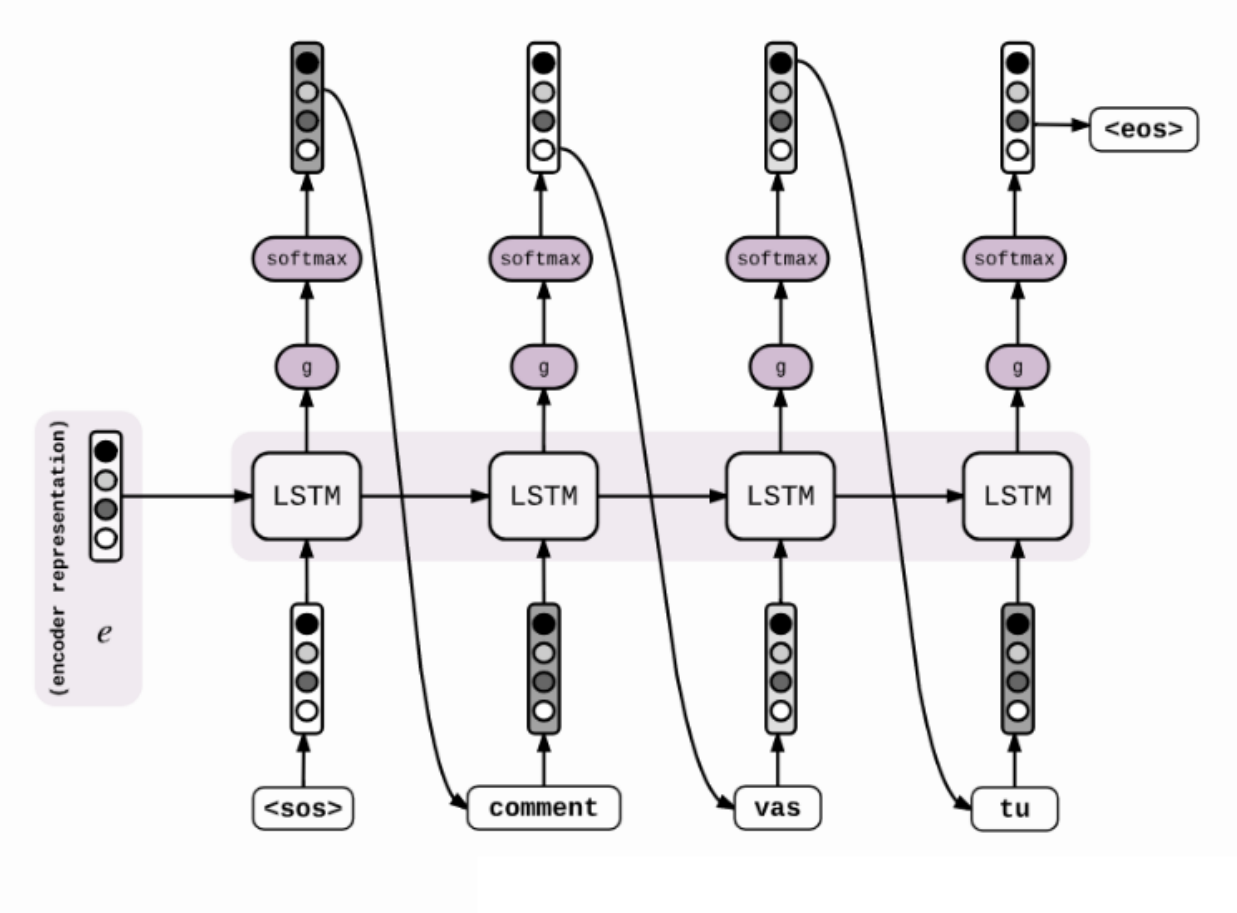
Seq2seq 改进



一个例子


e_0
e_1
e_2
LSTM
LSTM
LSTM
(encoder representation)
e
w_0
w_1
w_2
how
are
you

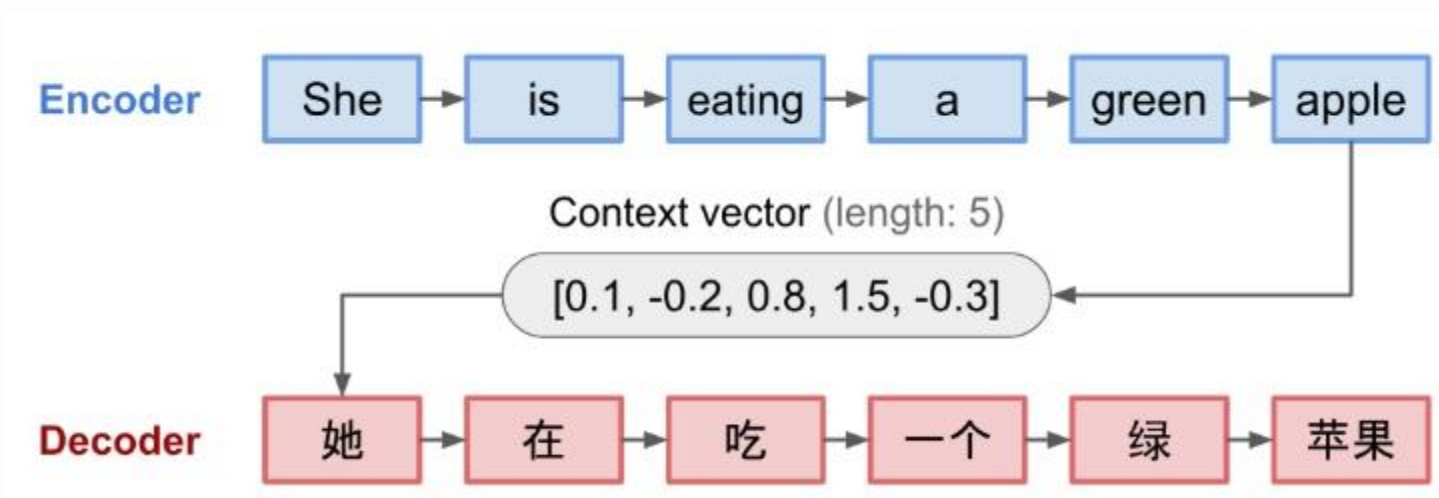
一个例子


(encoder representation)
e
LSTM
LSTM
LSTM
LSTM
g
g
g
g
softmax
softmax
softmax
softmax
<sos>
comment
vas
tu
<eos>

思考

- A B C->X Y Z
- C B A ->X Y Z

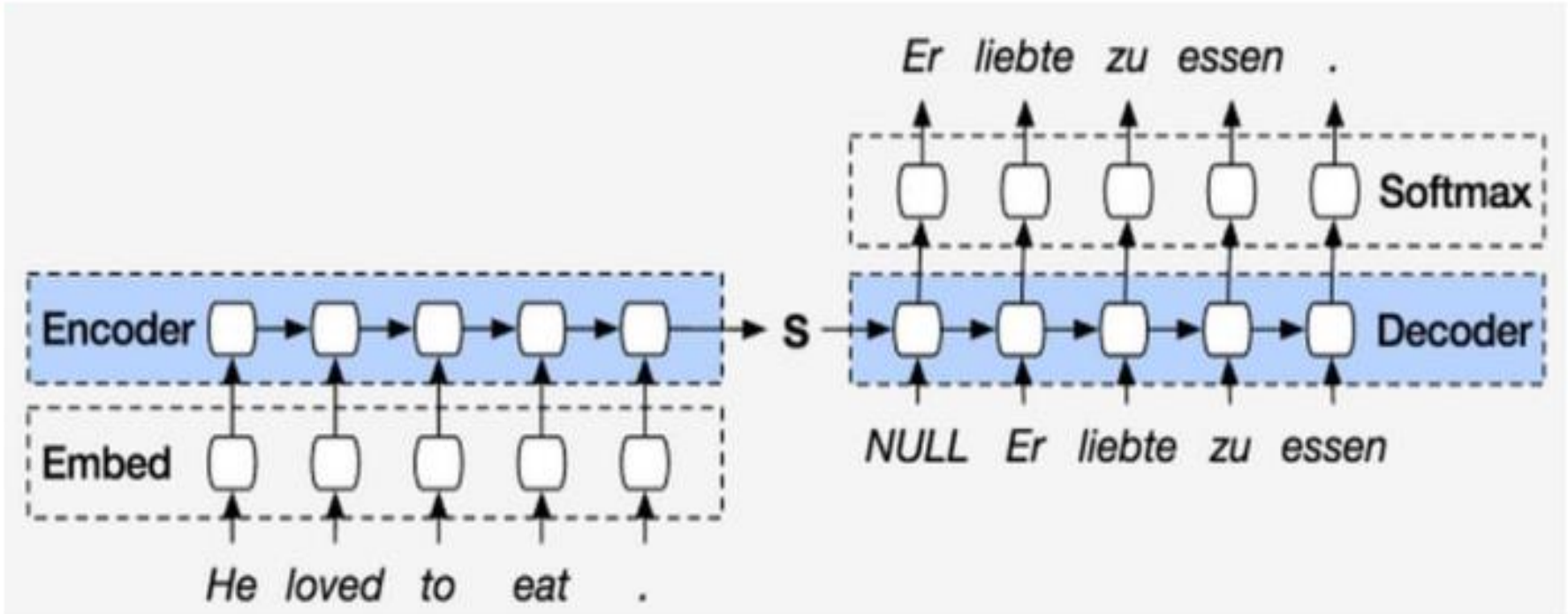
Seq2Seq问题所在



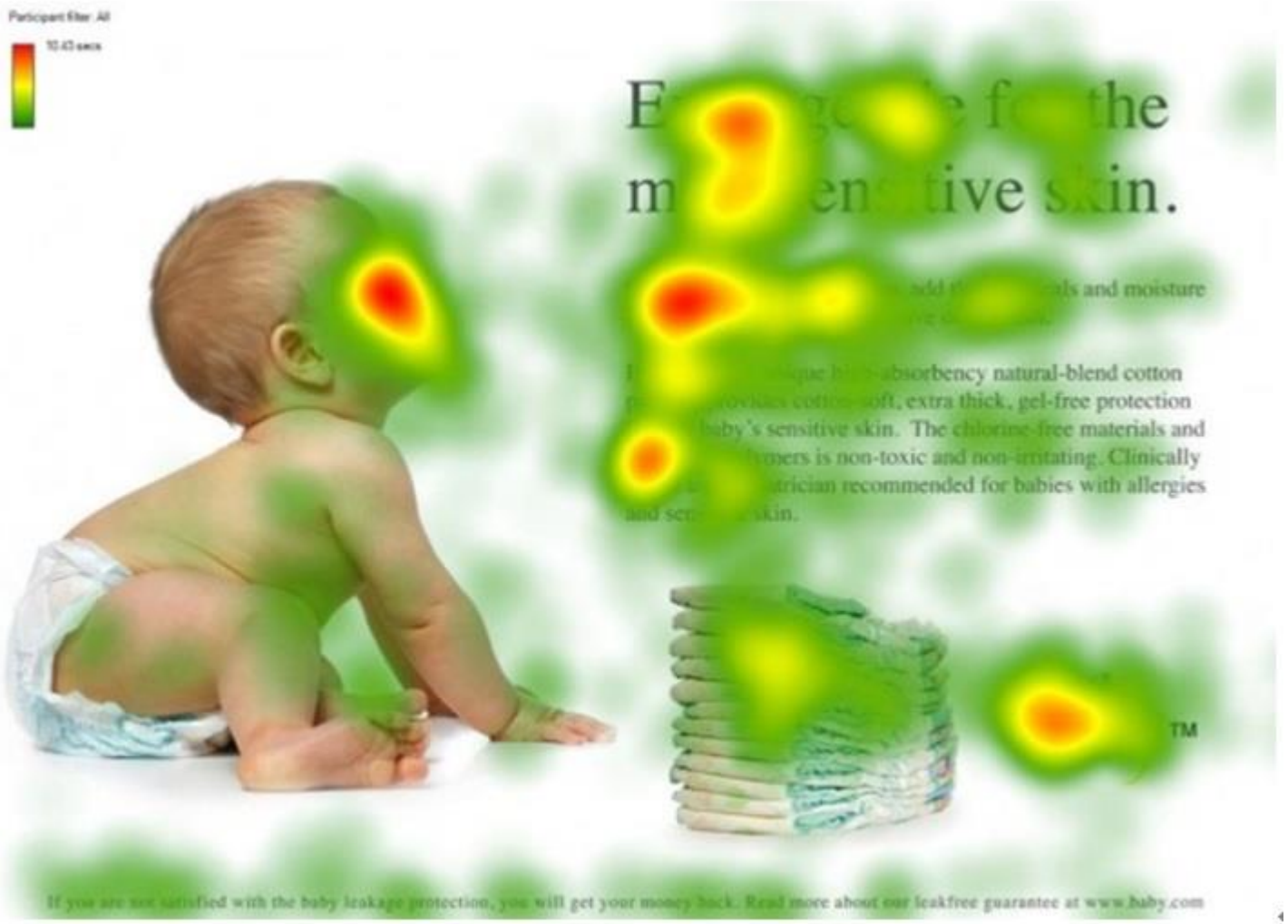
固定长度上下文向量具有一个明显的致命缺点——无法记忆长句子。一旦完成编码器输入序列的处理，就会遗忘开始的部分。因此注意力机制（Bahdanau et al.，2015）被提出，解决这个问题。

Seq2Seq问题所在

压缩损失了信息



Attention机制

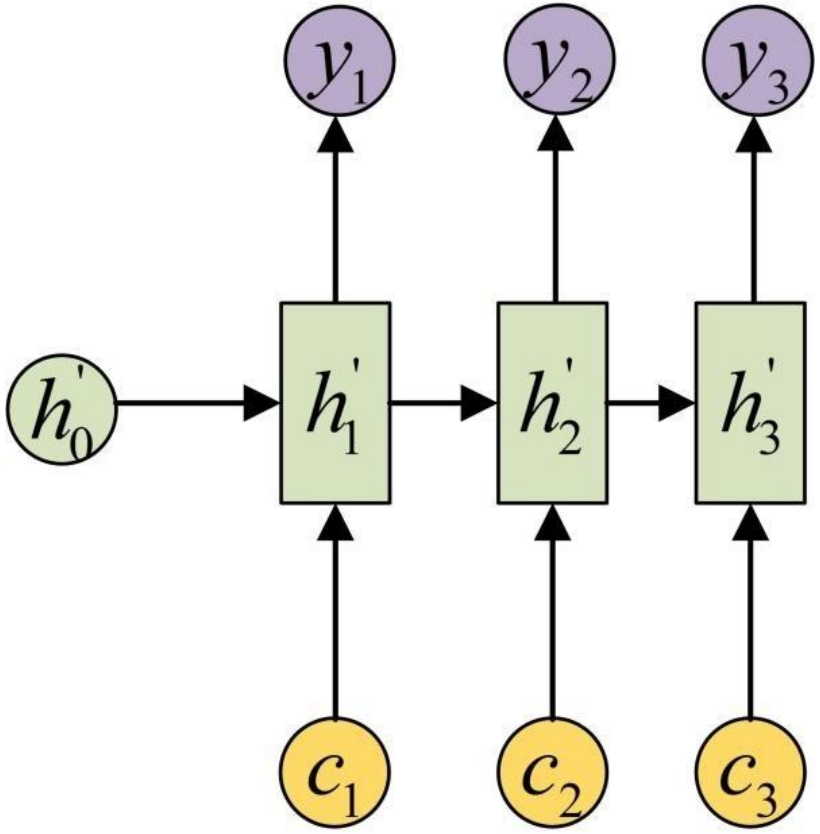


Attention机制

从这里可以看出，在生成目标句子的单词时，不论生成哪个单词，它们使用的输入句子Source的语义编码C都是一样的，没有任何区别。

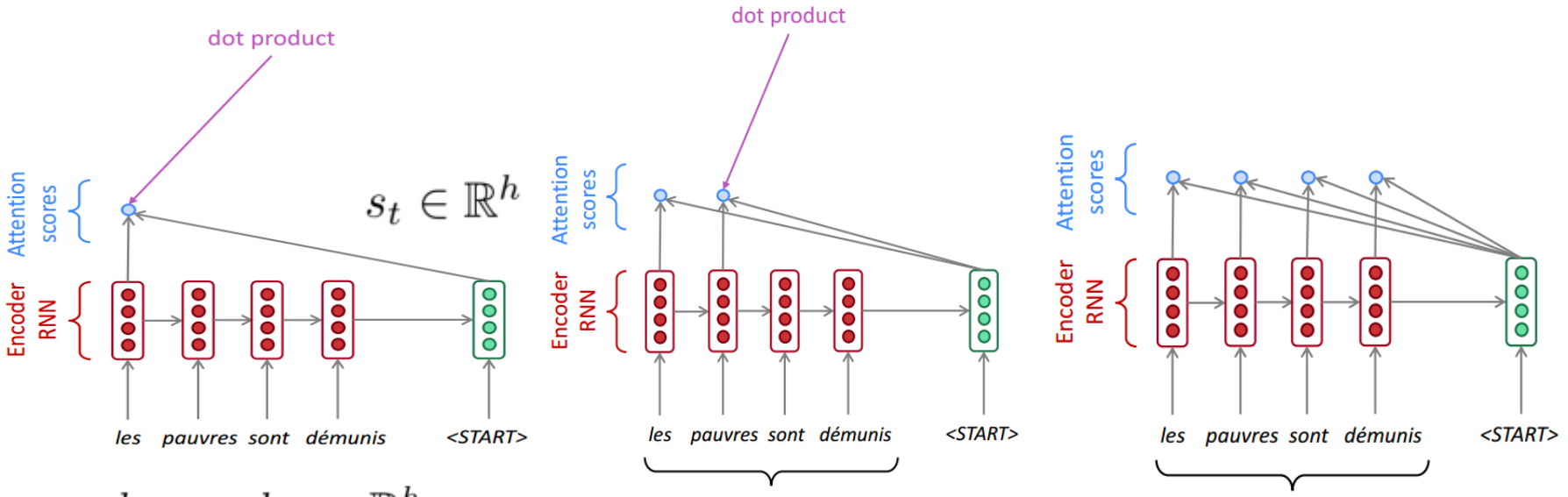
$$\mathbf{y_1} = \mathbf{f}(\mathbf{C})$$
$$\mathbf{y_2} = \mathbf{f}(\mathbf{C}, \mathbf{y_1})$$
$$\mathbf{y_3} = \mathbf{f}(\mathbf{C}, \mathbf{y_1}, \mathbf{y_2})$$

Attention机制



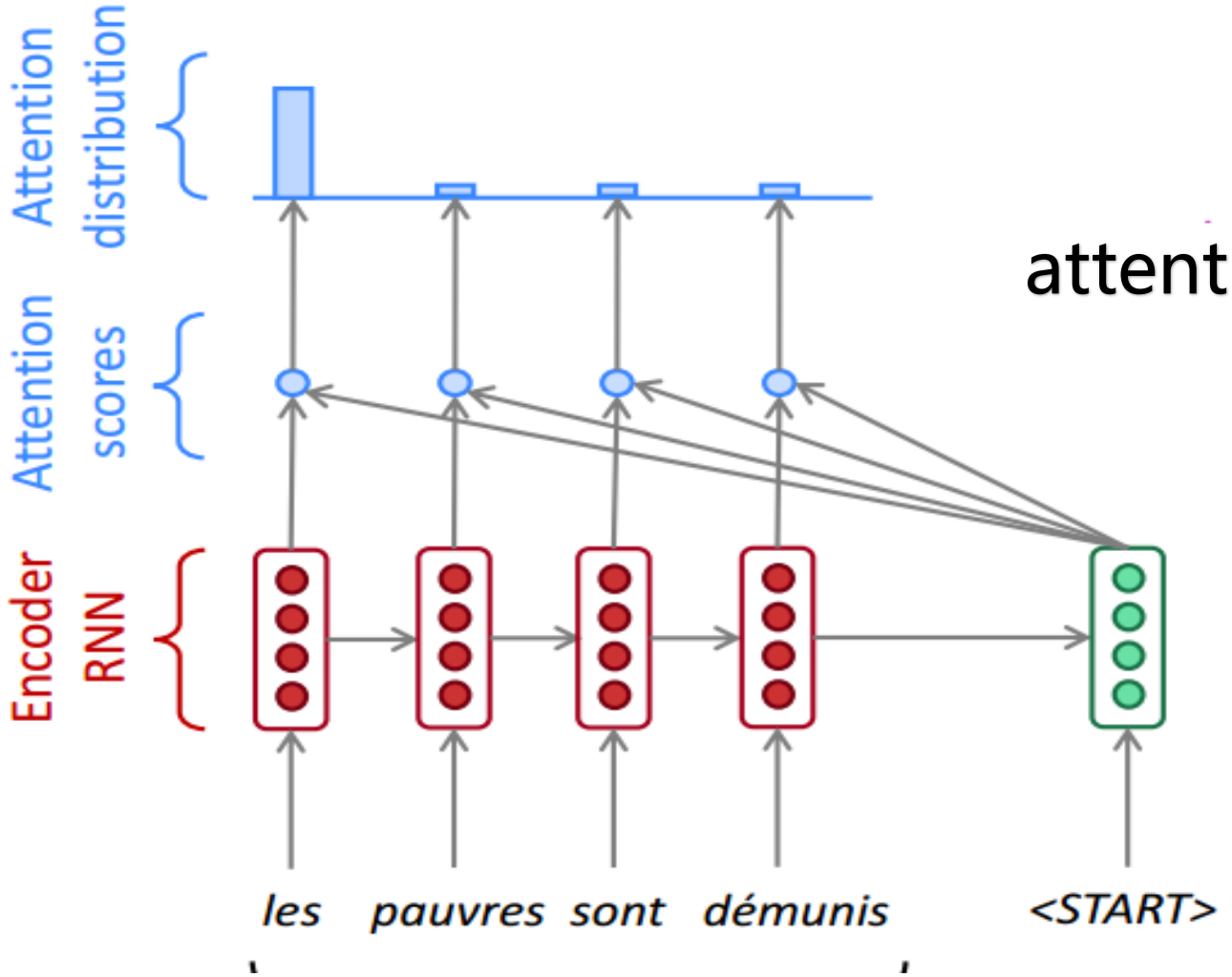
使用不同的C！！

$$y_1 = f1(C_1)$$
$$y_2 = f1(C_2, y_1)$$
$$y_3 = f1(C_3, y_1, y_2)$$



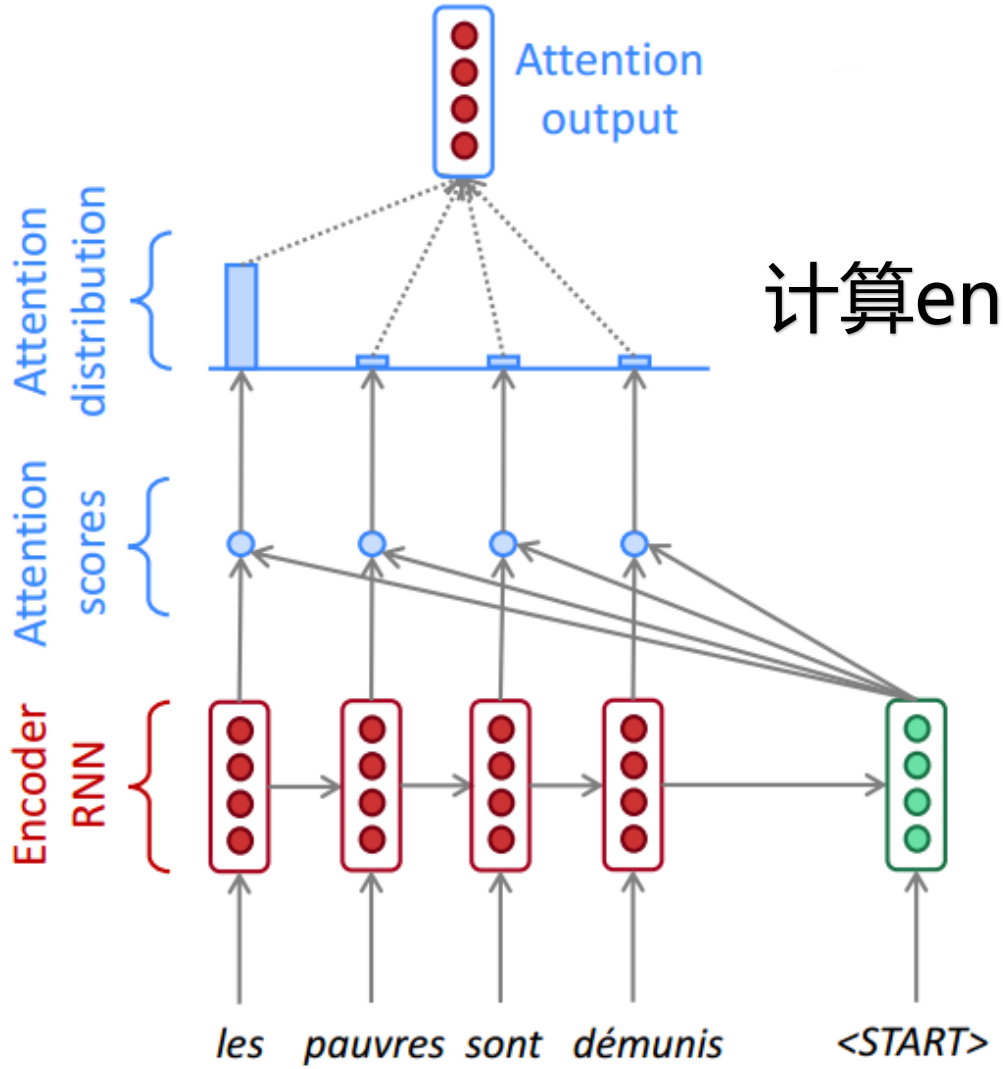
利用点积dot得到attention score

$$e^t = [s_t^T h_1, \ldots, s_t^T h_N]$$



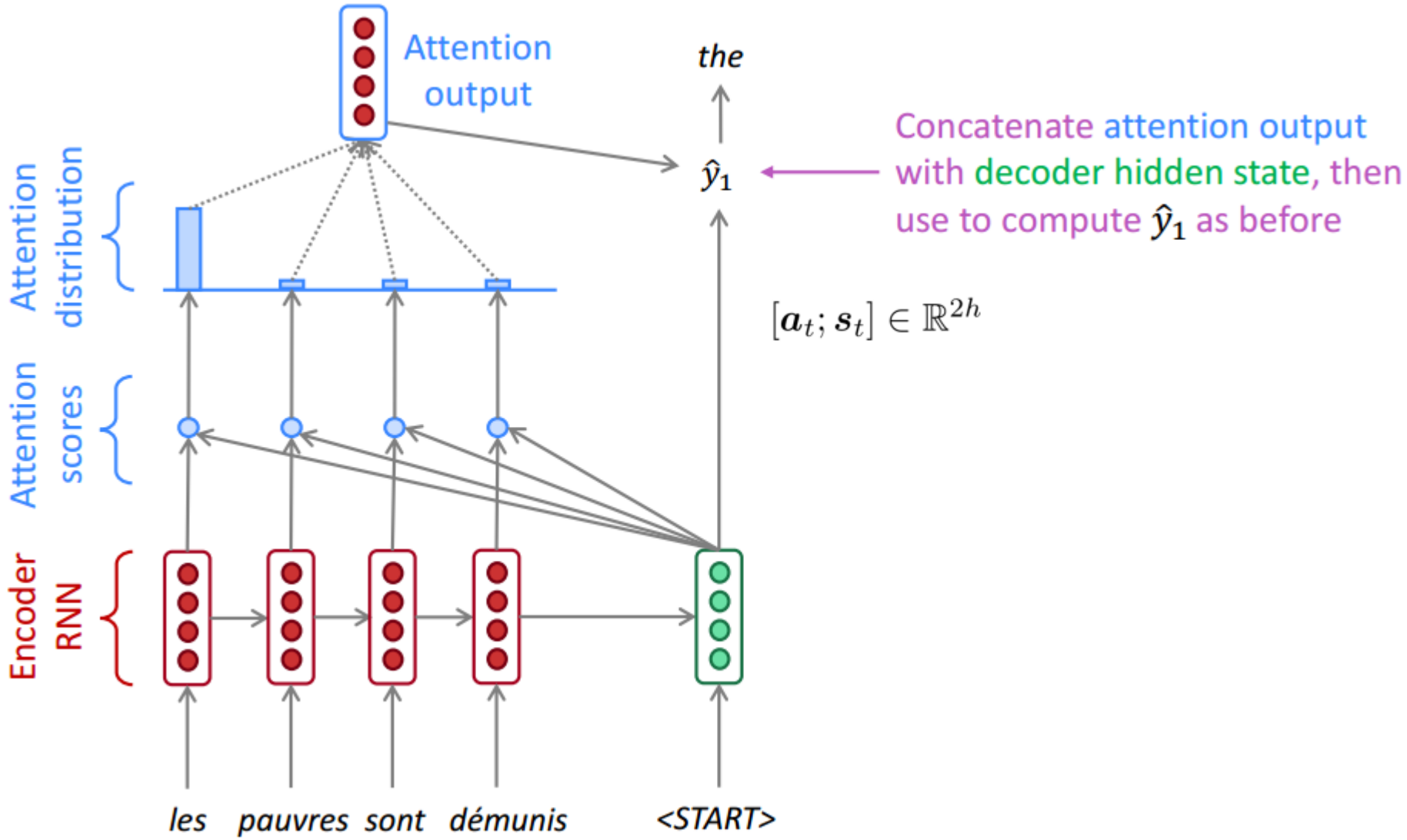
利用softmax函数：
attention scores转化为概率分布

$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

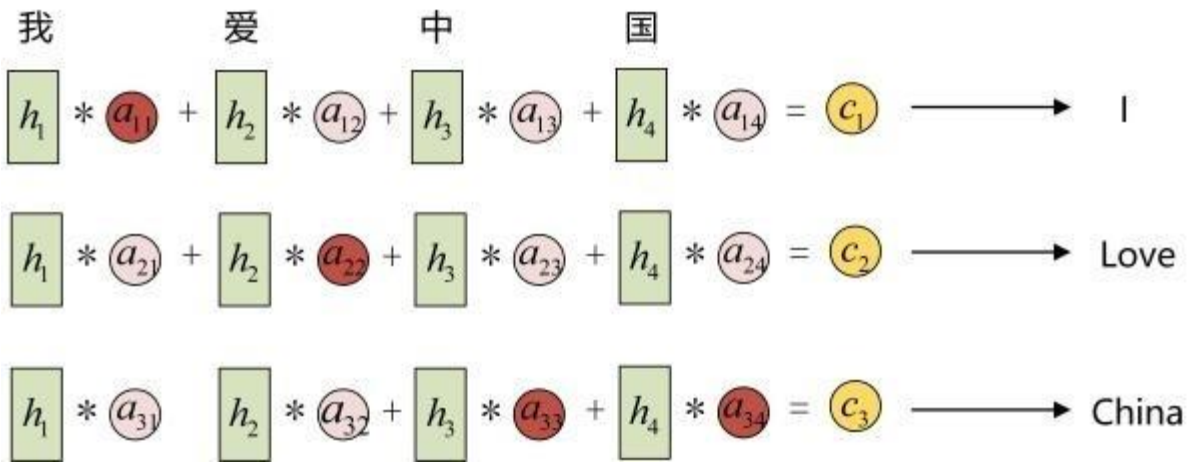


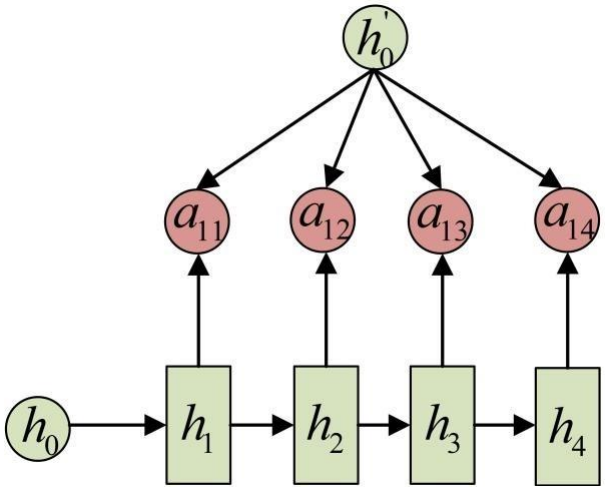
按照上一步概率分布：
计算encoder的hidden states的加权求和

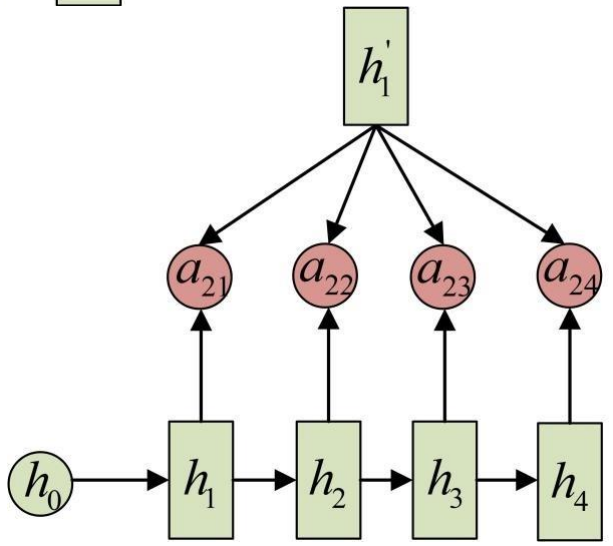
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

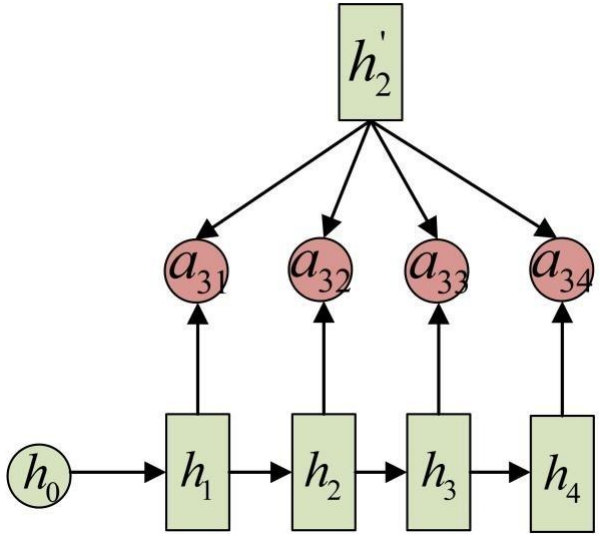

Attention output
the
Attention distribution
$\hat{y}_1$
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before
$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$
Attention scores
Encoder RNN
les
pauvres
sont
démunis
<START>

Attention机制（词对齐）









翻译中的注意力机制

How old are you ?

5	5	5	80	5	你
80	5	10	5	0	多
10	80	10	0	0	大
5	10	65	15	5	了
0	0	10	0	90	?

attention的一个通用定义

- 按照Stanford大学课件上的描述，attention的通用定义如下：给定一组向量集合values，以及一个向量query，
- attention机制是一种根据该query计算values的加权求和的机制。attention的重点就是这个集合values中的每个value的“权值”的计算方法。
- 有时候也把这种attention的机制叫做query的输出关注了（或者说叫考虑到了）原文的不同部分。（Query attends to the values）

不同的score求出方式

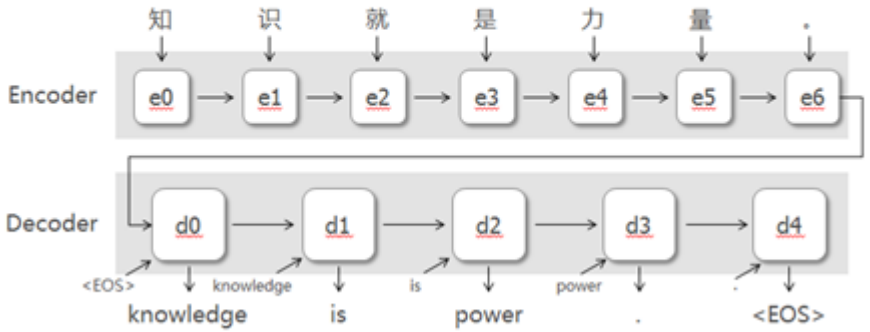
$$a_t = align(m_t, m_s) = \frac{\exp(f(m_t, m_s))}{\sum_{s'} \exp(f(m_t, m_{s'}))}$$

$$f(m_t, m_s) = \begin{cases} m_t^\top m_s & \text{dot} \\ m_t^\top W_a m_s & \text{general} \\ W_a [m_t; m_s] & \text{concat} \\ v_a^\top \tanh(W_a m_t + U_a m_s) & \text{perceptron} \end{cases}$$

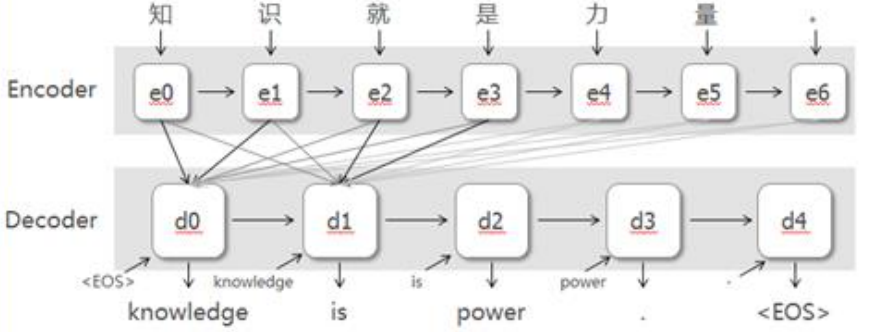
Attention机制（more…）

Background: Neural Machine Translation

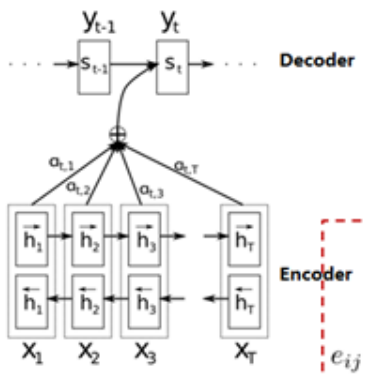
- Sequence to Sequentce (encoder–decoder)



Attention-based Neural Machine Translation



Learning to Align and Translate



$$p(y_i|y_1,\ldots,y_{i-1},\mathbf{x}) = g(y_{i-1}, s_i, c_i)$$

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

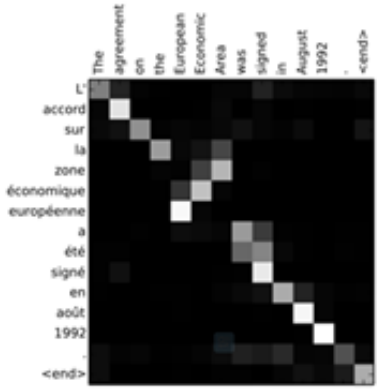
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j) = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

Results

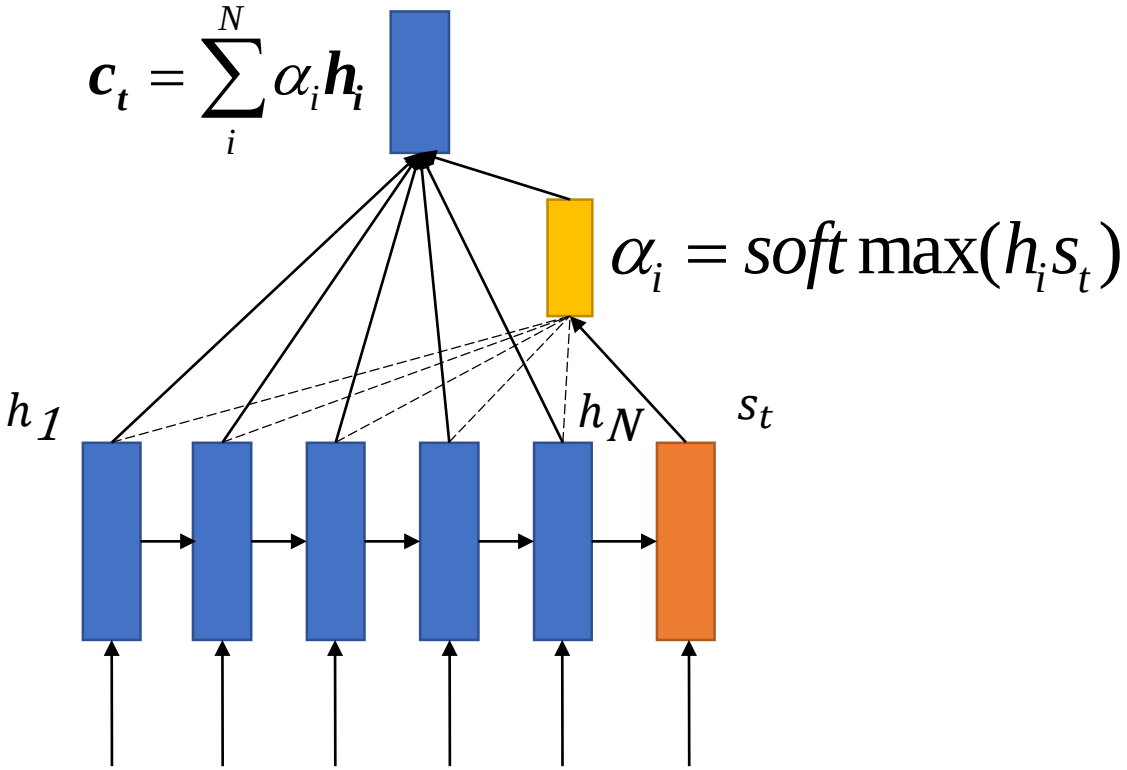
- Alignment
- Long Sentences

Model	All	No UNK°
RNNenc-30	13.93	24.19
RNNsearch-30	21.50	31.44
RNNenc-50	17.82	26.71
RNNsearch-50	26.75	34.16



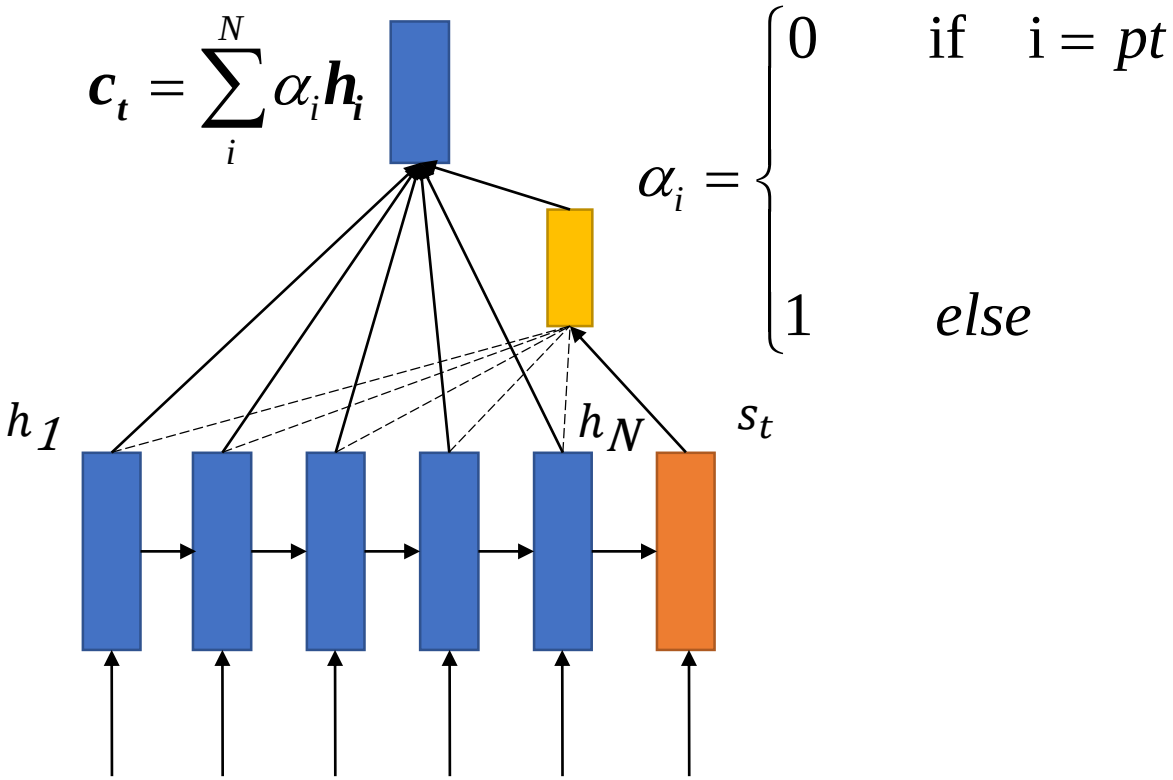
Soft attention

Soft attention就是我们上面讲过的那种最常见的attention，是在求注意力分配概率分布的时候，对于输入句子X中任意一个单词都给出个概率，是个概率分布

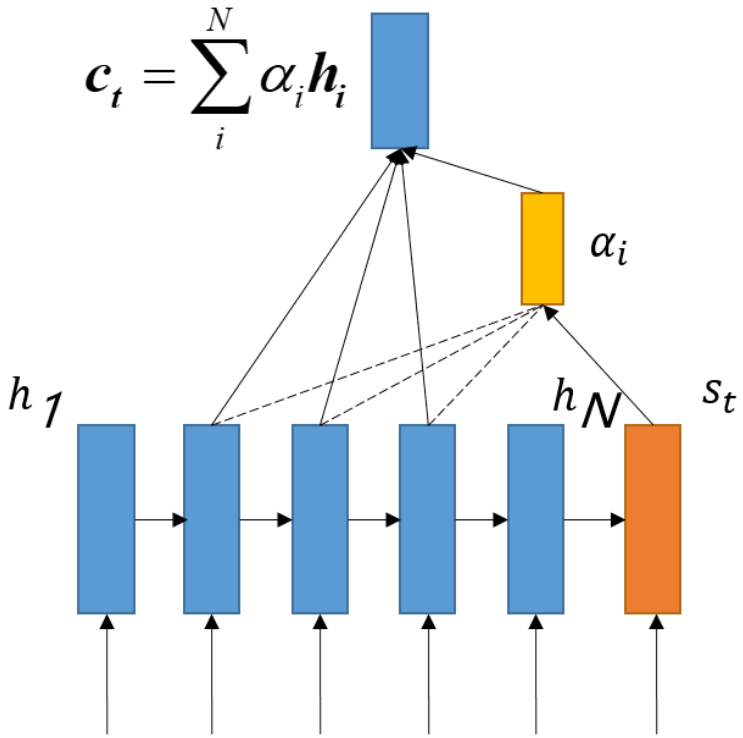


Hard attention

Soft是给每个单词都赋予一个单词match概率，那么如果不这样做，直接从输入句子里面找到某个特定的单词，然后把目标句子单词和这个单词对齐，而其它输入句子中的单词硬性地认为对齐概率为0，这就是Hard Attention Model的思想。



local attention （半软半硬attention）



寻找pt并计算alpha的方式又大致分为两种：
Local – m：假设对齐位置就是pt = t （线性对齐）
然后计算窗口内的softmax，窗口外的alpha可以取0

$$\alpha_i = \frac{exp(score(\tilde{h}_i, s_t))}{\sum_i exp(score(\tilde{h}_i, s_t))}$$

Local – p：先通过一个函数预测pt在[0,S]之间，然后取一个类高斯分布乘以softmax。

$$pt = S \cdot sigmoid(v_p{}^T tanh(W_p s_t))$$

$$\alpha_t = \frac{exp(score(\tilde{h}_i, s_t))}{\sum_i exp(score(\tilde{h}_i, s_t))} exp(-\frac{(s - pt)}{2\sigma^2})$$

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

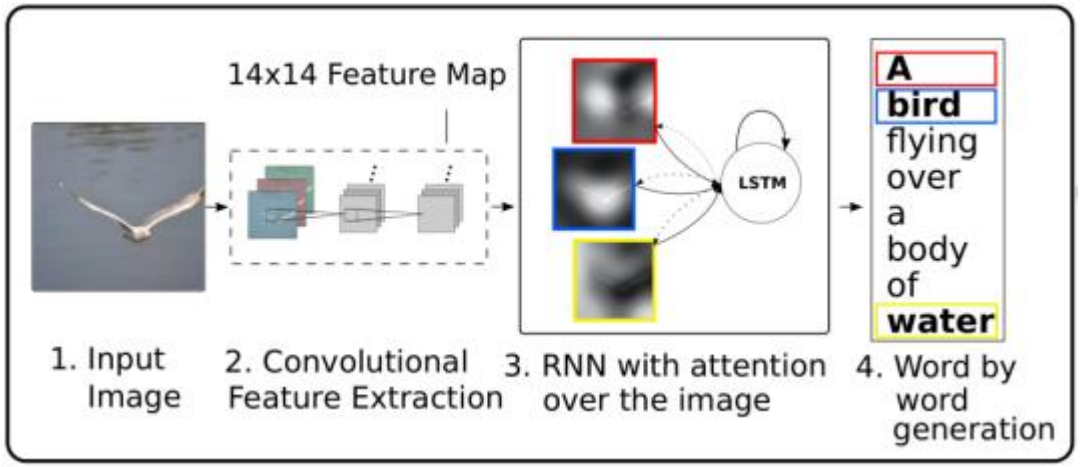
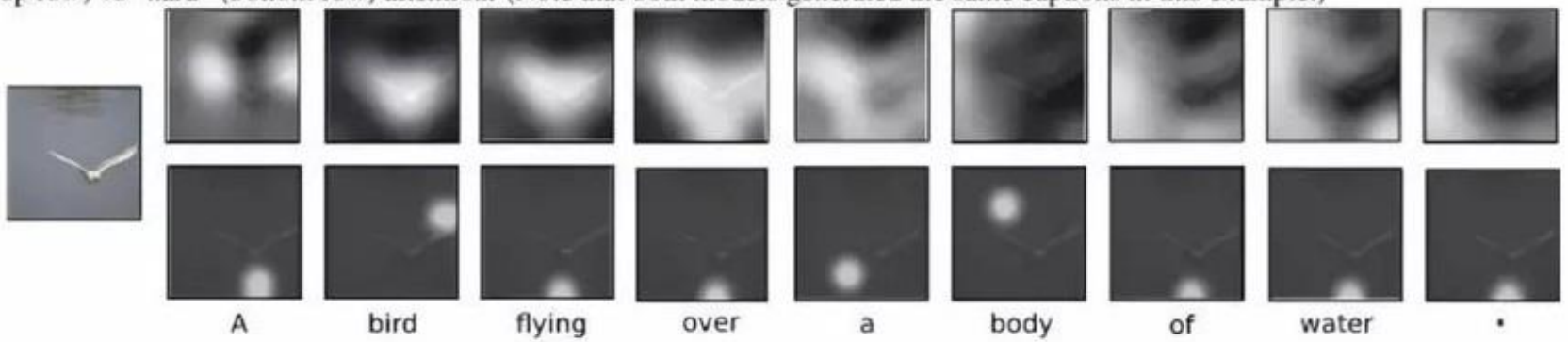


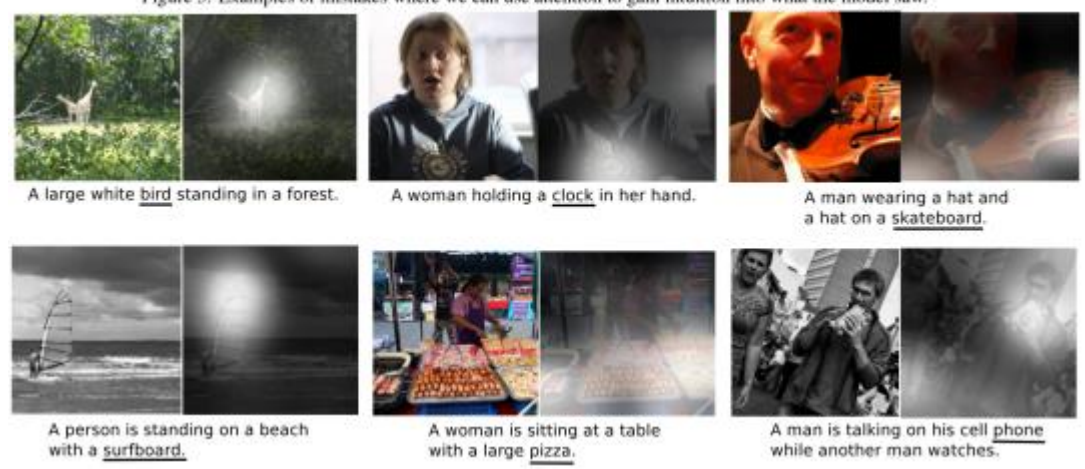
Figure 2. Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)



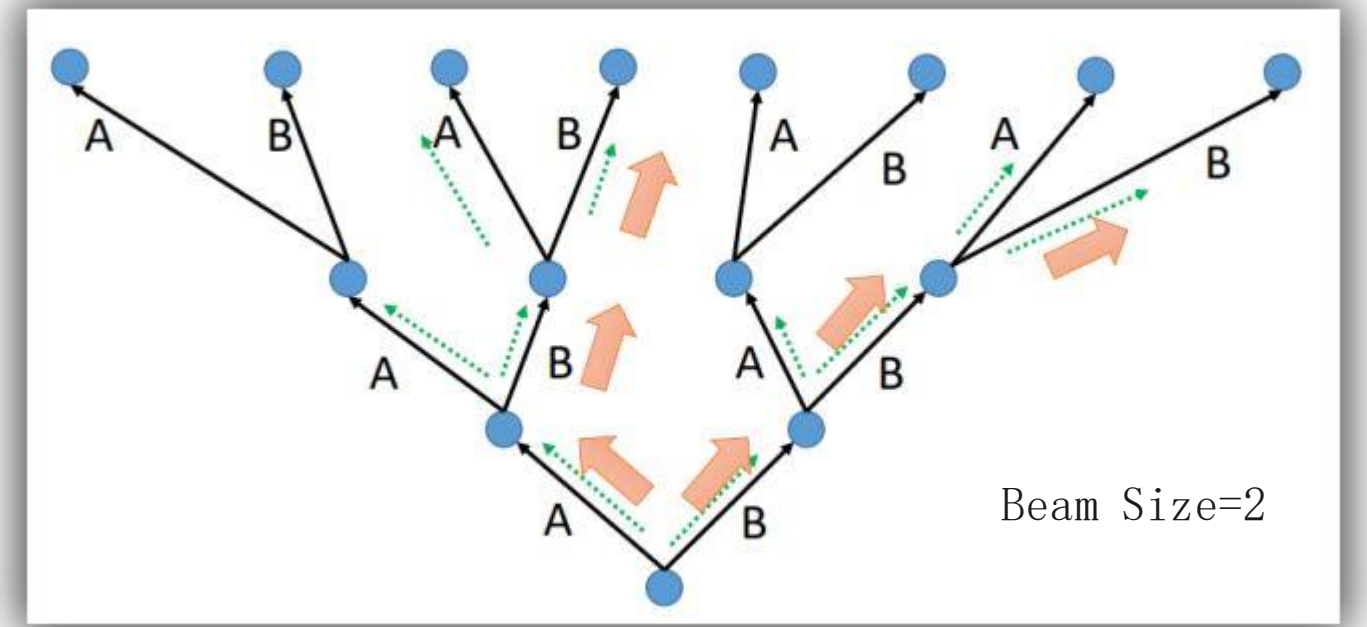
Show, Attend and Tell: Neural Image Caption Generation with Visual Attention

Figure 4. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

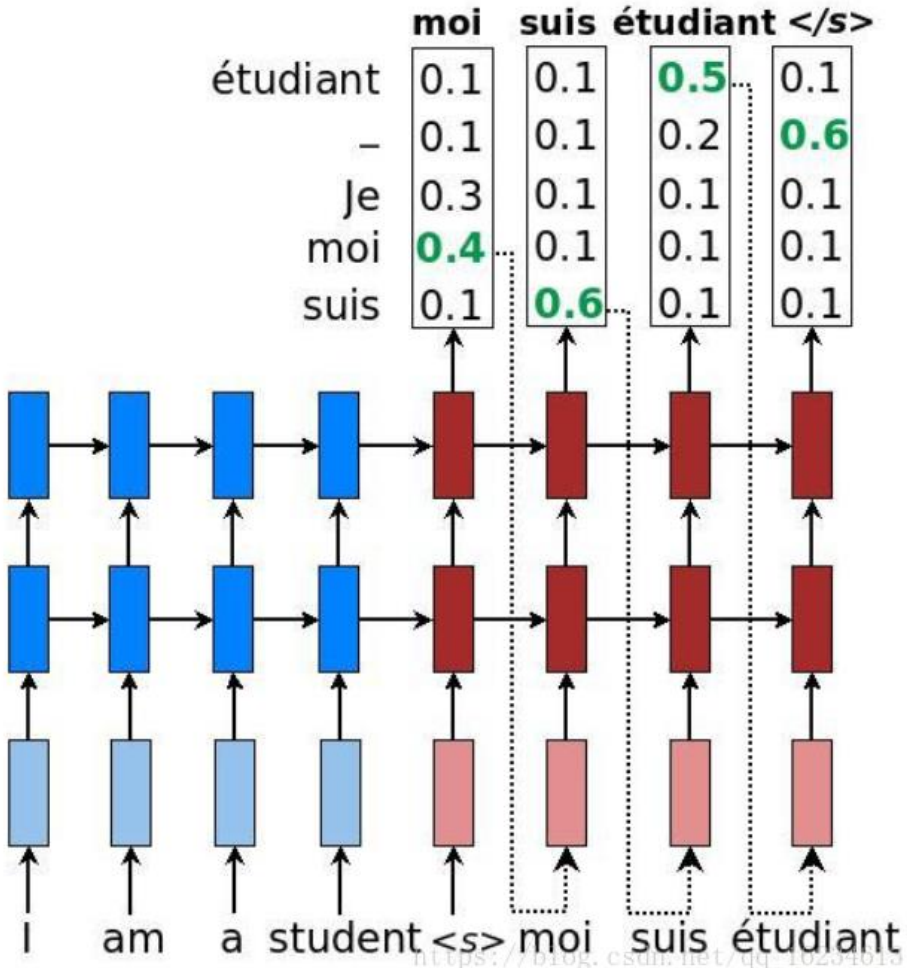
Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



集束搜索



贪心搜索vs集束搜索



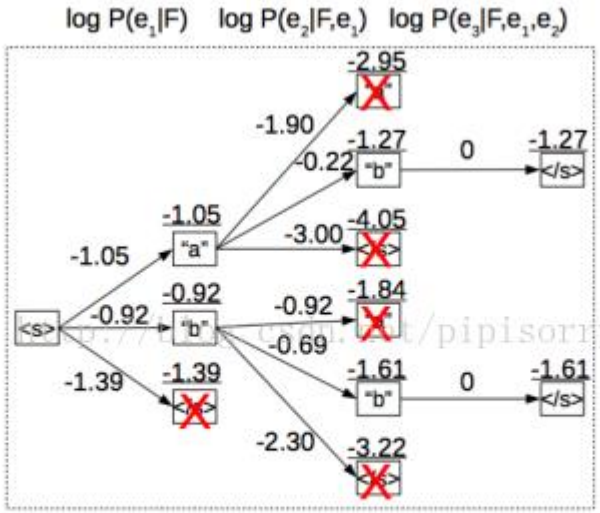
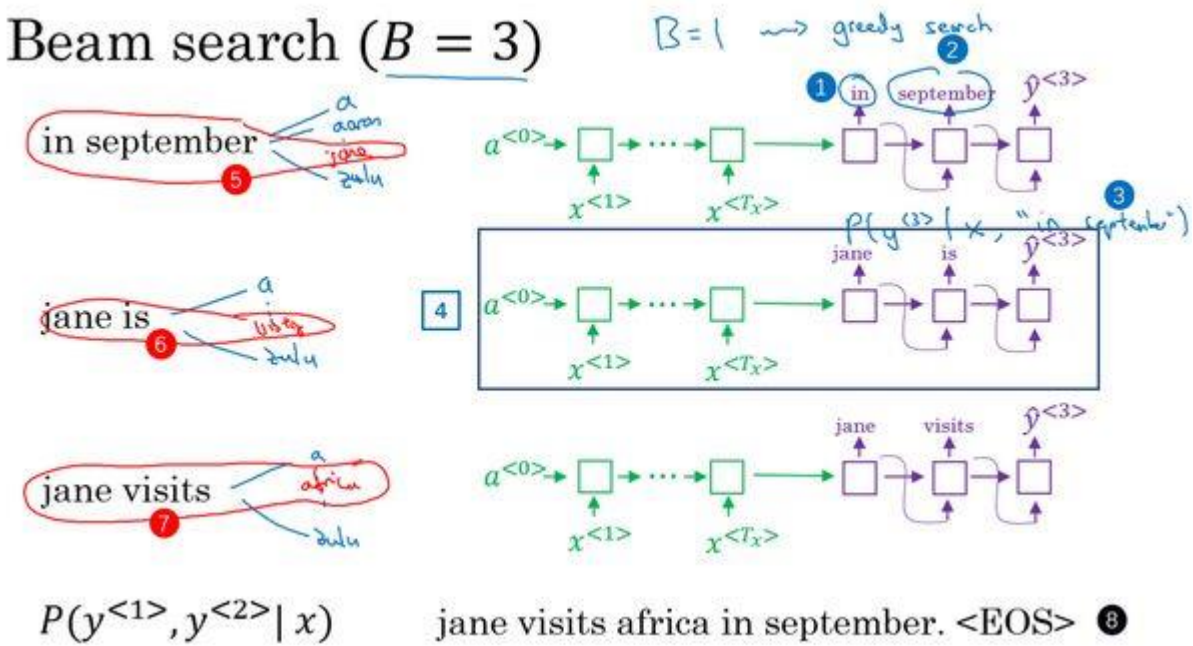


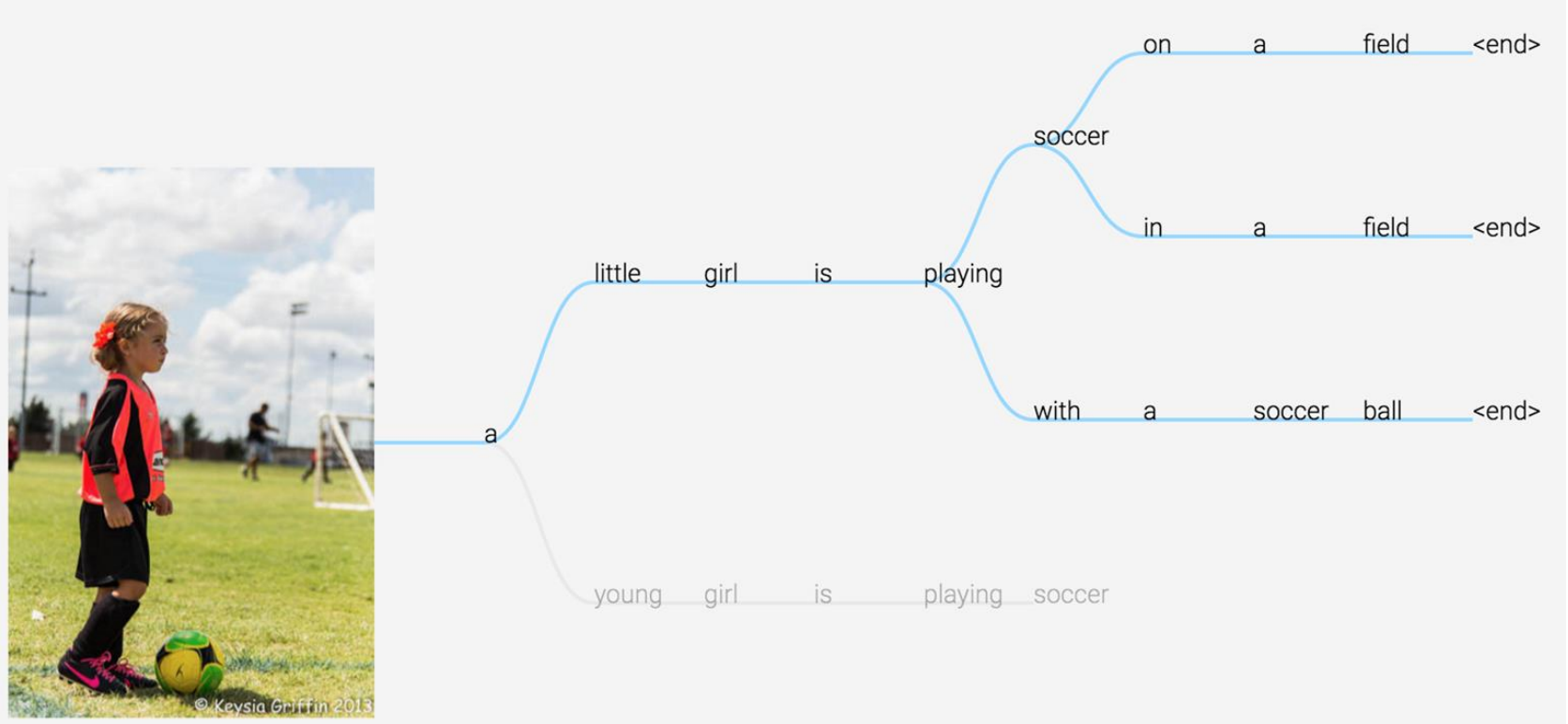
Figure 23: An example of beam search with $b = 2$. Numbers next to arrows are log probabilities for a single word $\log P(e_t \mid F, e_1^{t-1})$, while numbers above nodes are log probabilities for the entire hypothesis up until this point.

集束搜索

法语句子"Jane visite l'Afrique en septembre."
翻译1-Jane is visiting Africa in September.
翻译2-Jane is going to be visiting Africa in September.



集束搜索


a
little girl is playing
soccer
on a field <end>
in a field <end>
with a soccer ball <end>
young girl is playing soccer
© Keysia Griffin 2013