NLP项目

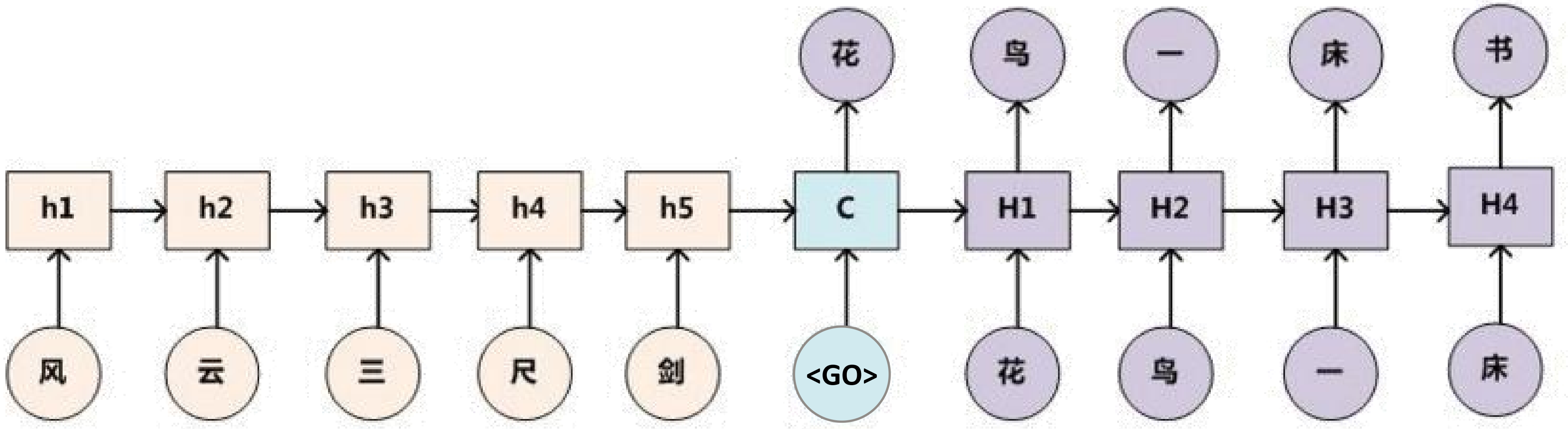
Attention

课程内容

- Attention 结构讲解
- Seq2Seq+Attention

Seq2Seq问题

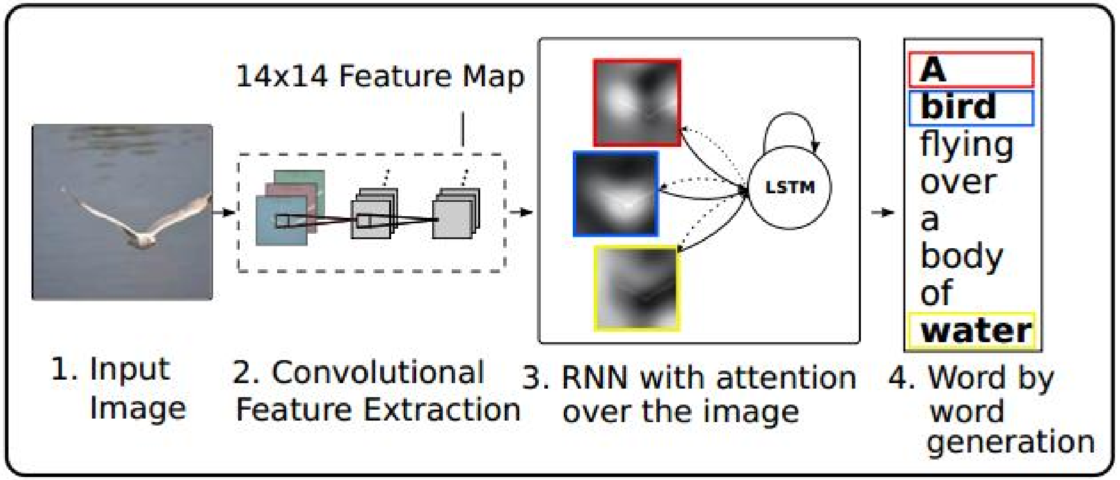
- 问题描述：”风”对应的特征对于下联的影响是最弱的。

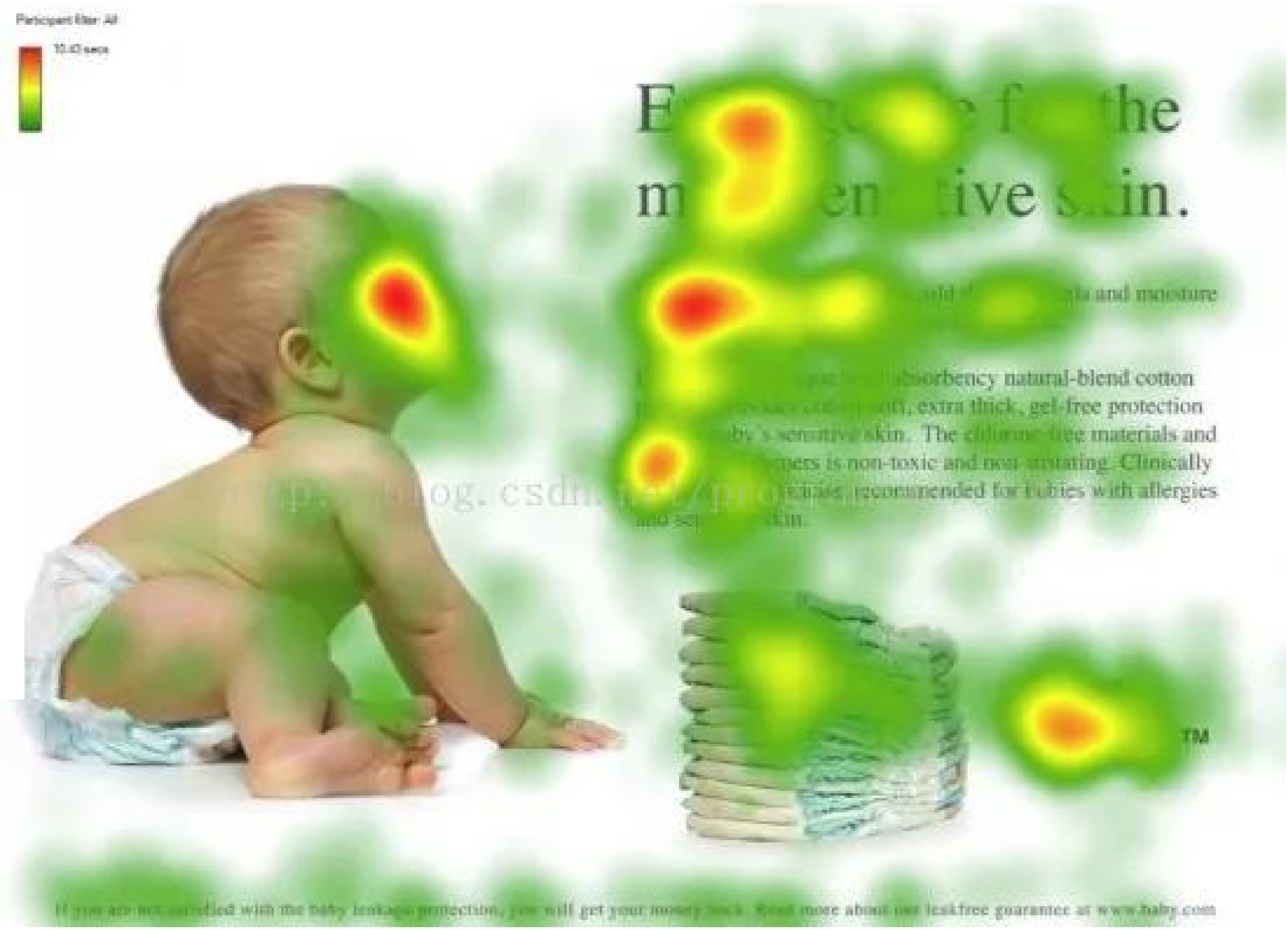


下联生成的时候，不仅仅需要考虑上联的整个语义文本信息，针对下联里面的每个token(字)都需要和上联的对应位置的token(字)进行"关联"

1. 字句对等；
2. 词性对品；
3. 结构对应；
4. 节律对拍；
5. 平仄对立；
6. 形对意联；

Attention


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation



Seq2Seq Attention

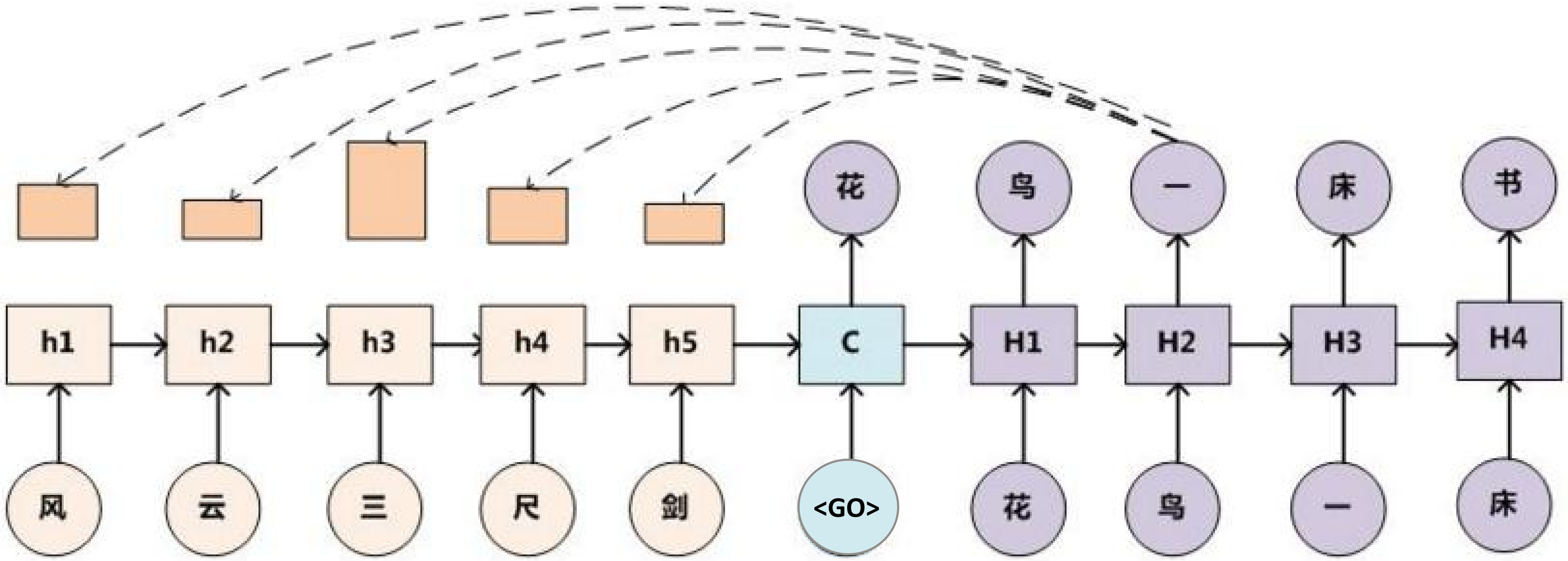
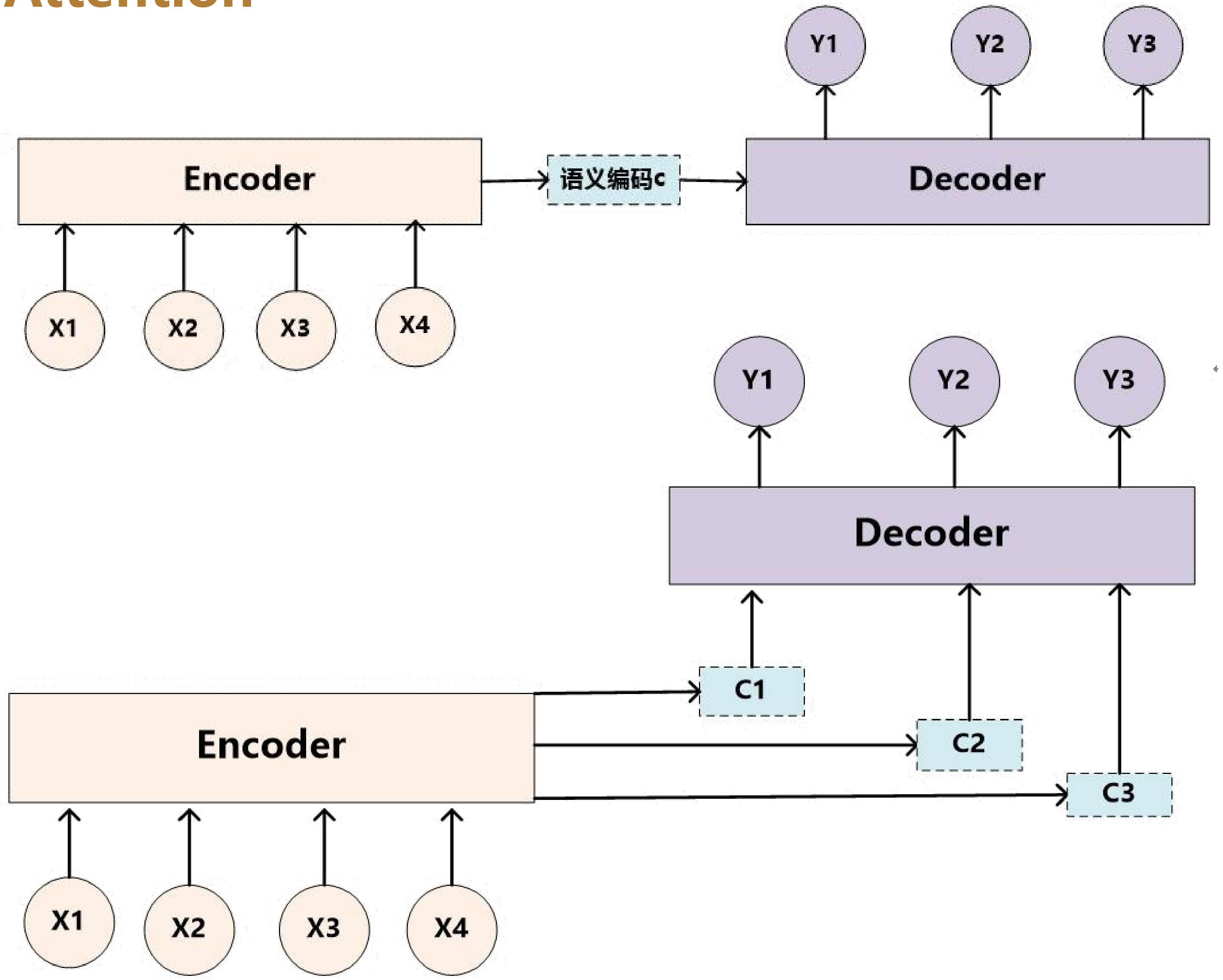
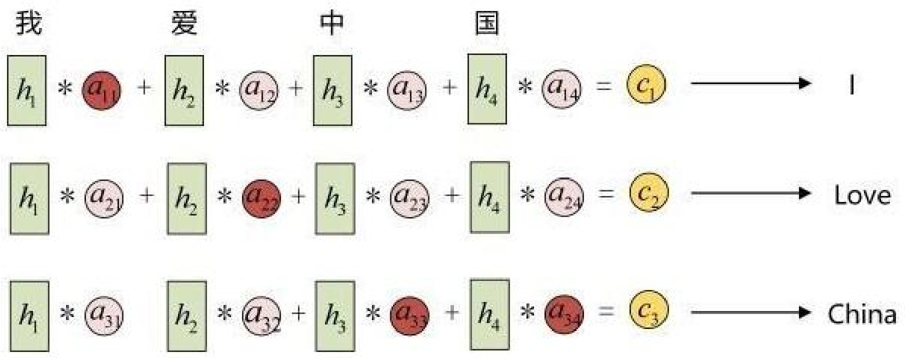


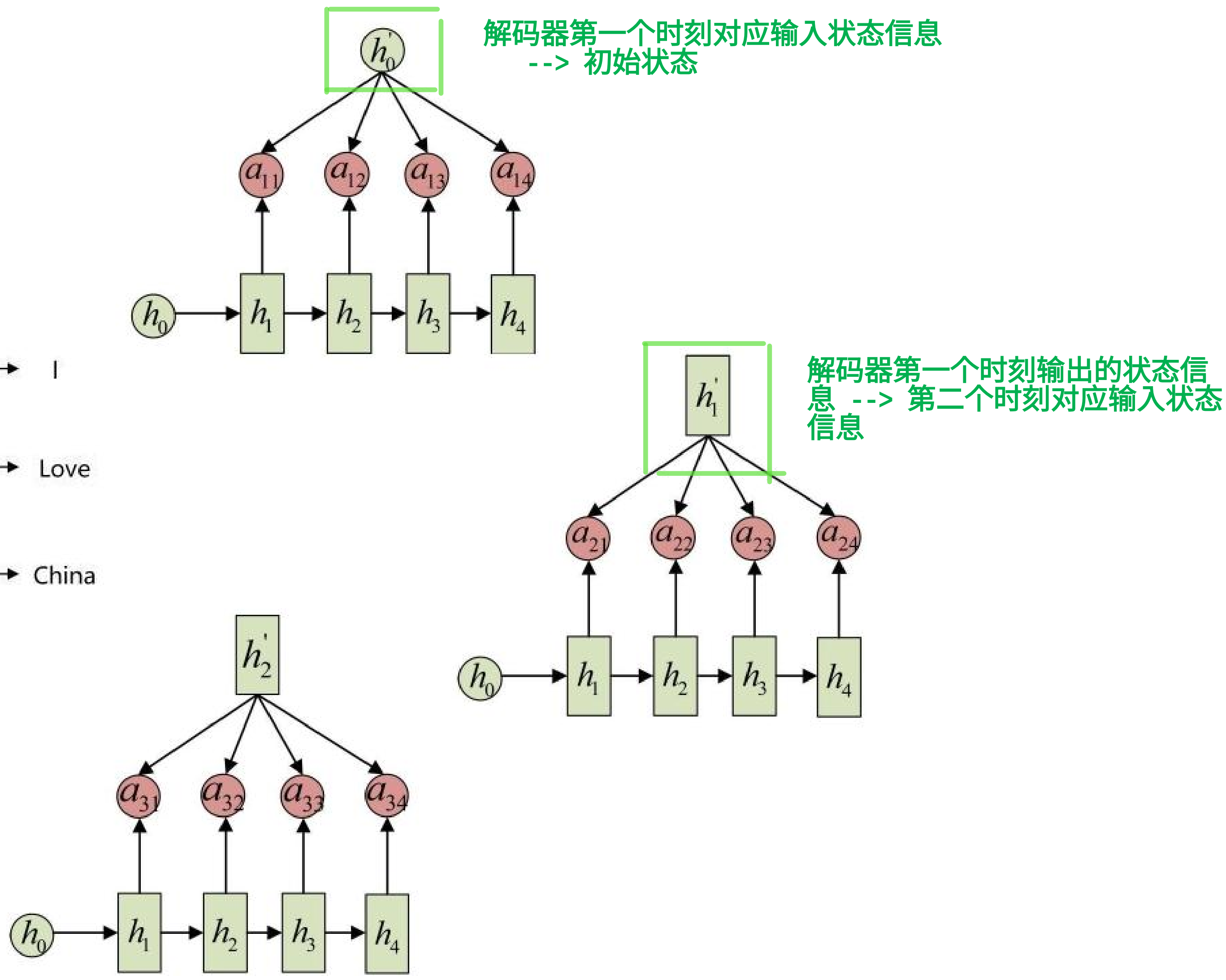
图3. Attention模型

Seq2Seq Attention

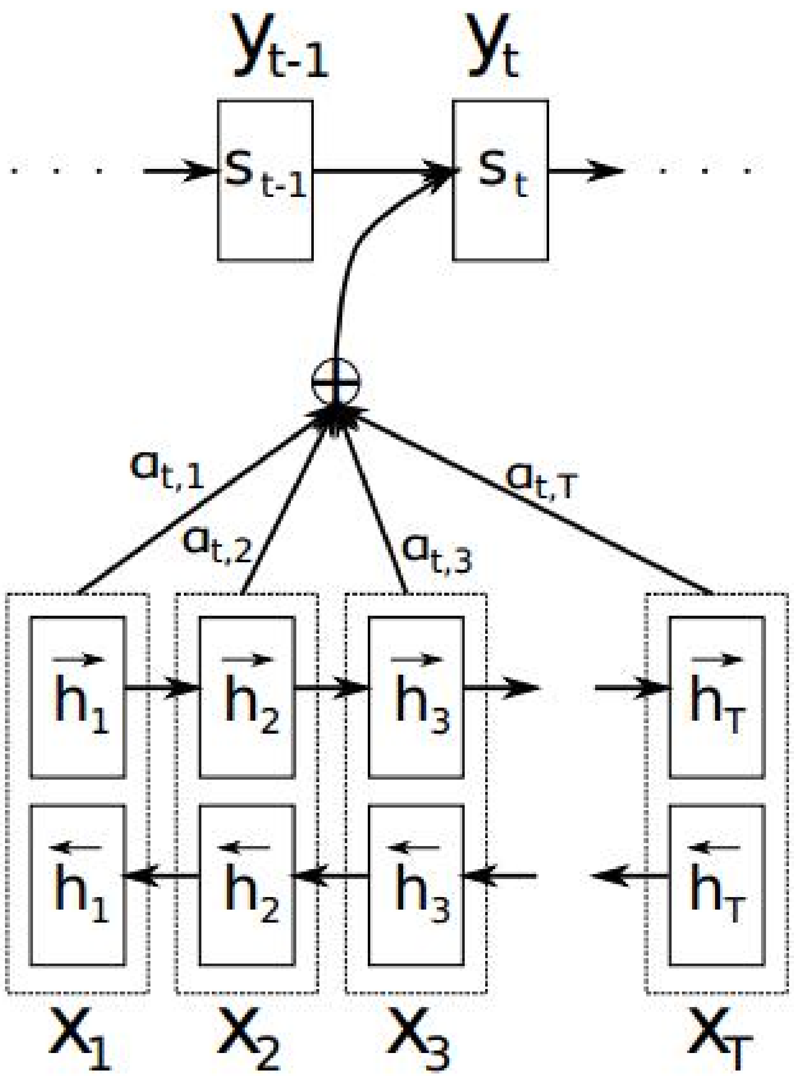


Attention对齐机制(词对齐)

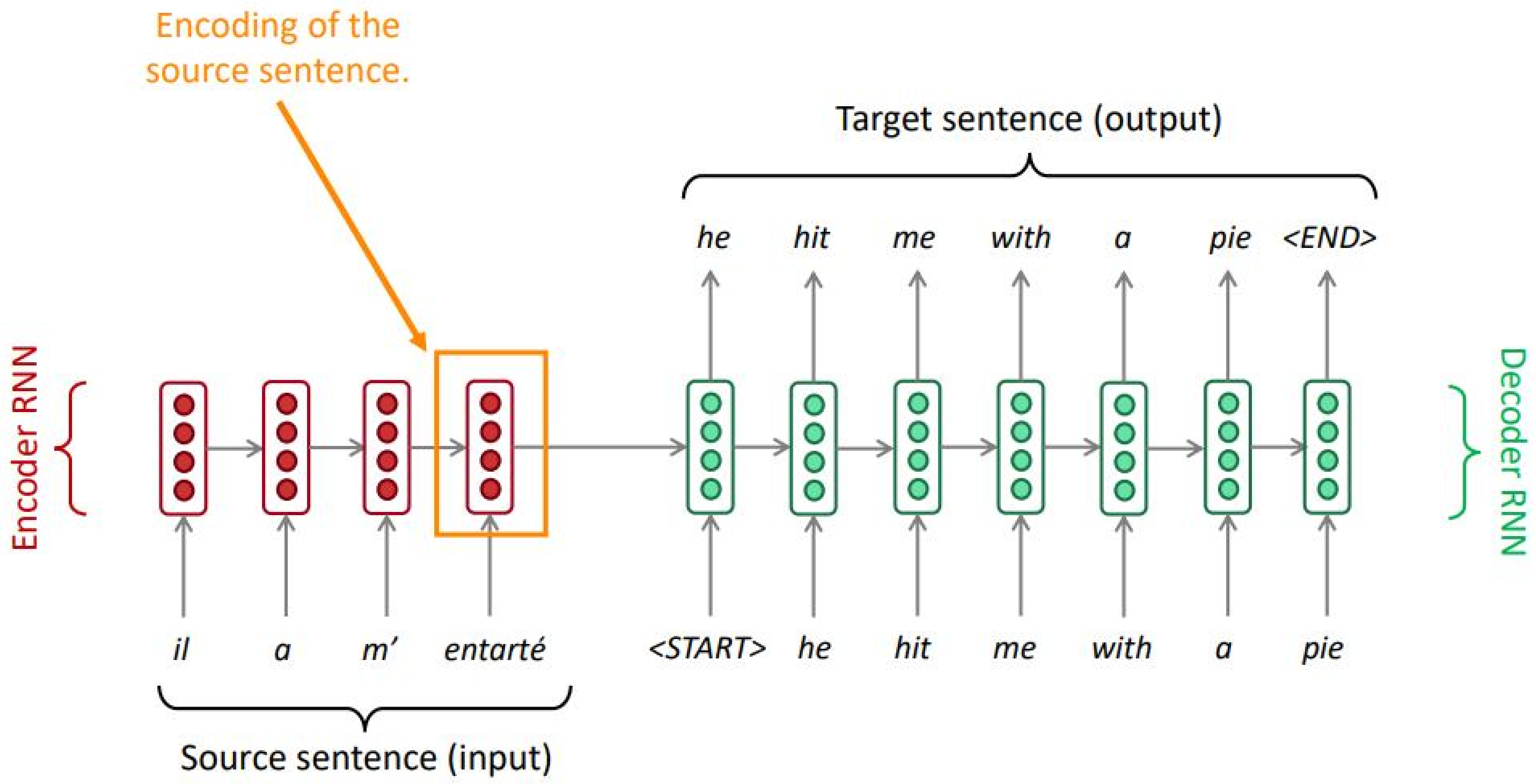




Seq2Seq Attention

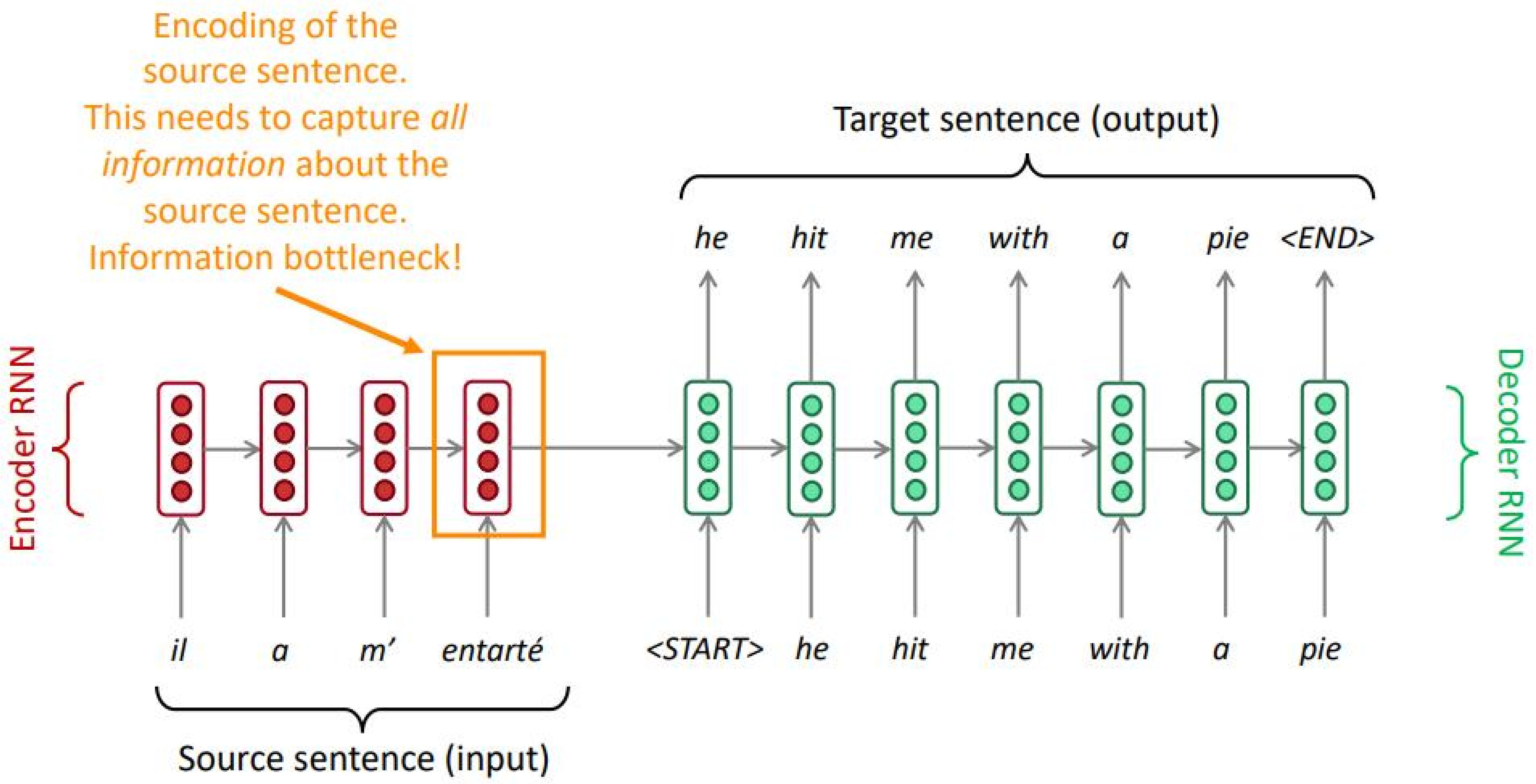


Seq2Seq Attention计算过程

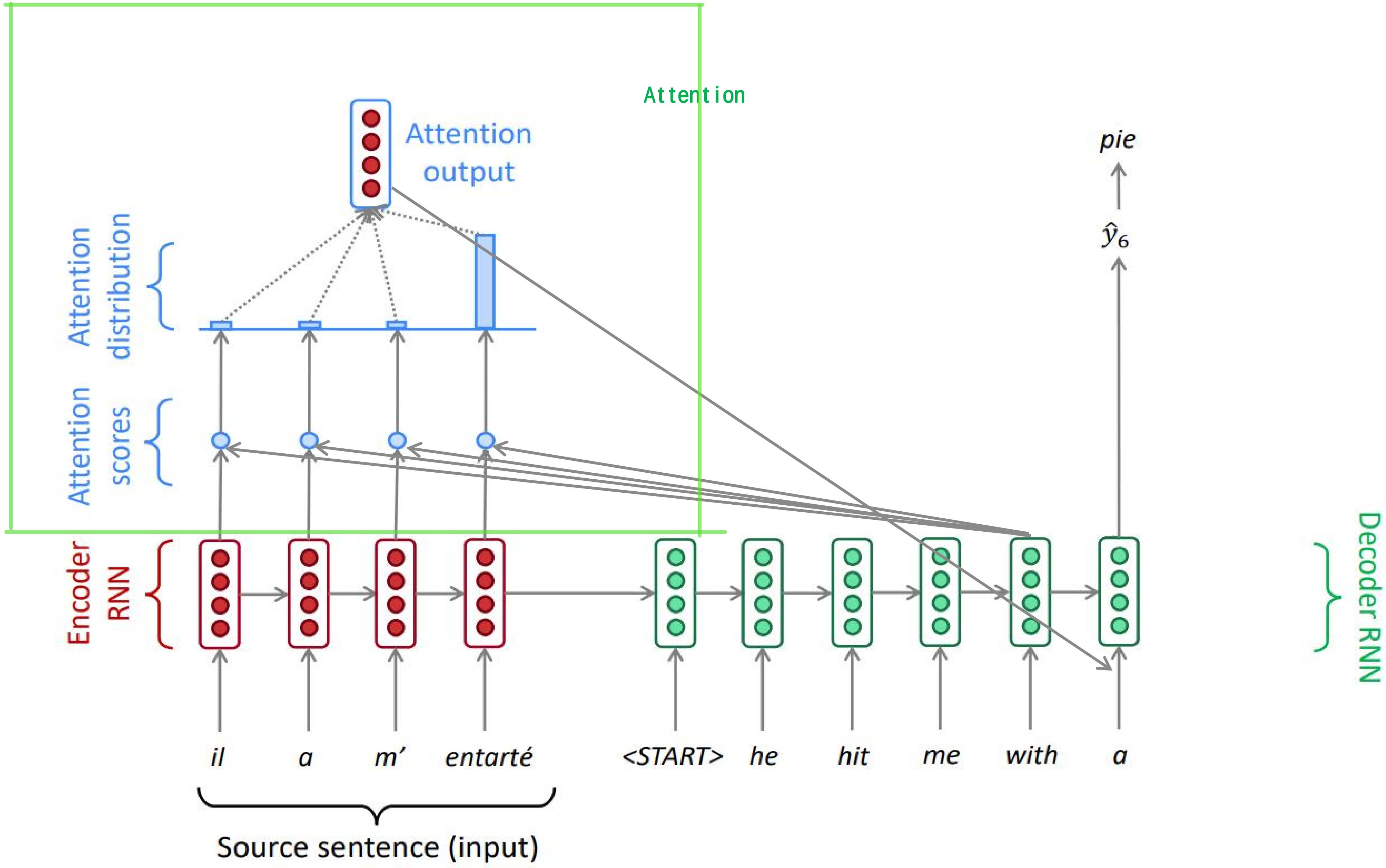


Problems with this architecture?

Seq2Seq Attention计算过程



Seq2Seq Attention计算过程



Seq2Seq Attention计算过程

- Encoder hidden states/output values：h_i; 总Encoder时刻n个。
- 时刻t，Decoder hidden states: s_t;
- 基于每个时刻Encoder输出以及上一个时刻Decoder的状态来构建Attention Scores：

$$e_{t,i} = F\left(h_i, s_{t-1}\right) \quad e_t = \left(e_{t,1}, e_{t,2}, \dots, e_{t,n}\right)$$

Attention

- 对e进行softmax转换，得到概率分布：$\alpha_t = softmax\left(e_t\right)$
- 基于概率分布以及所有Encoder的输出计算出Attention值；$a_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$
- 将Decoder当前时刻的输入和Attention值结合形成新的输入数据，然后进行普通的RNN Decoder操作。

$$y'_t = \left[y_t; a_t\right]$$

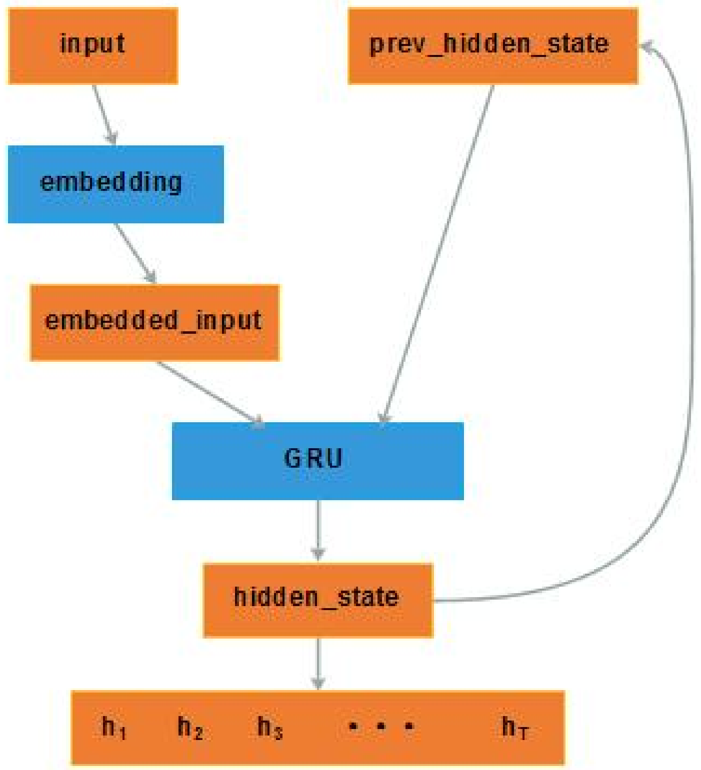
Seq2Seq Attention计算过程

- Attention Scores的计算函数F在不同论文中有很多形式，主要方式如下：
  - 乘法Attention：
    $$e_{t,i} = s_{t-1}^{T} h_i$$
    $$\boxed{e_{t,i} = s_{t-1}^{T} h_i / \sqrt{d}}$$
    现有的框架中应用最多的方式
  - 加法Attention：
    $$e_{t,i} = s_{t-1}^{T} W h_i$$
    $$\boxed{e_{t,i} = u^{T} \tanh\left(W_1 h_i + W_2 s_{t-1}\right)}$$
    $$e_{t,i} = W_1 h_i + W_2 s_{t-1}$$
    $$e_{t,i} = W h_i$$

TensorFlow默认

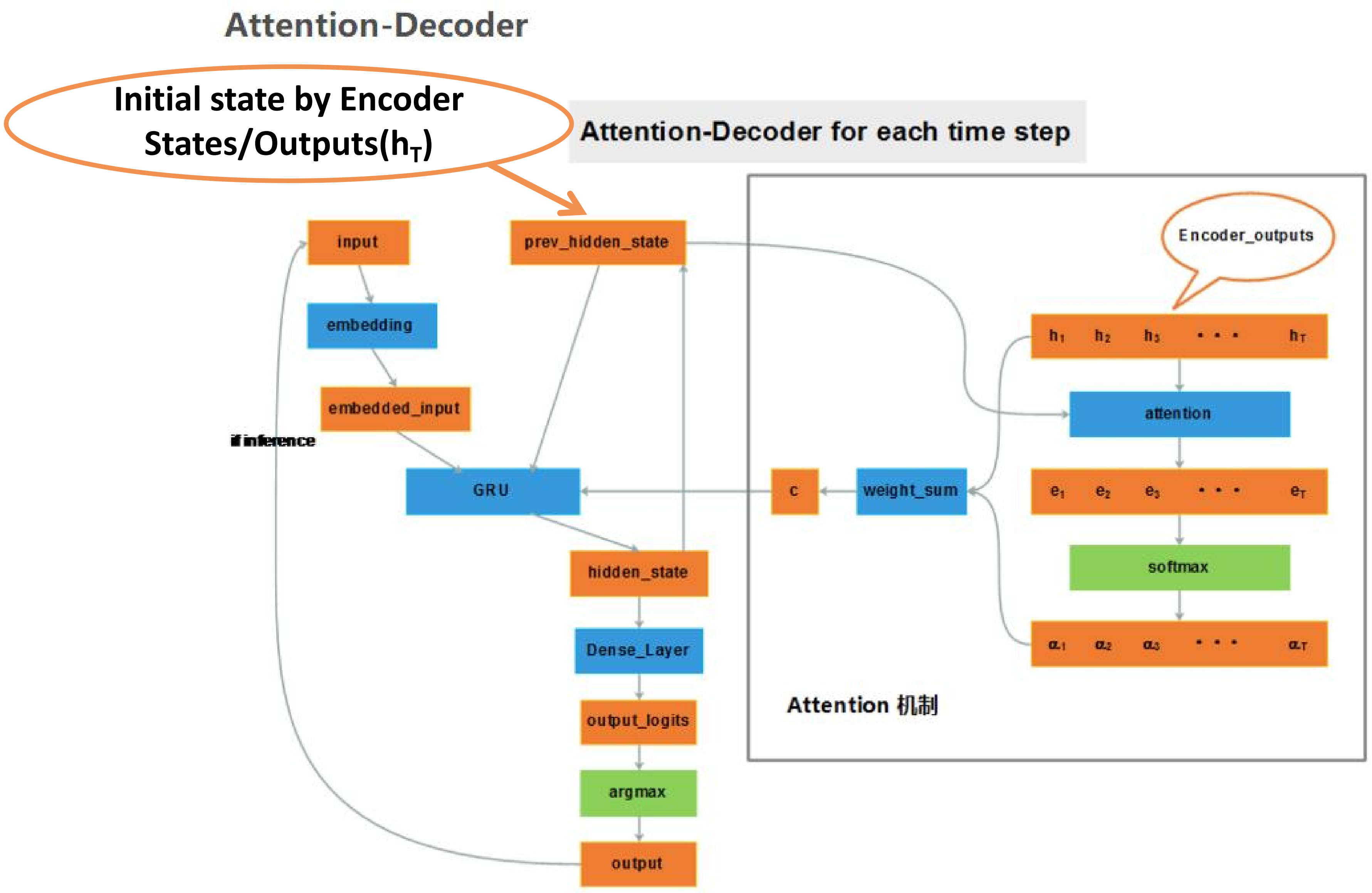
Seq2Seq Attention计算过程

Bi-RNN Encoder



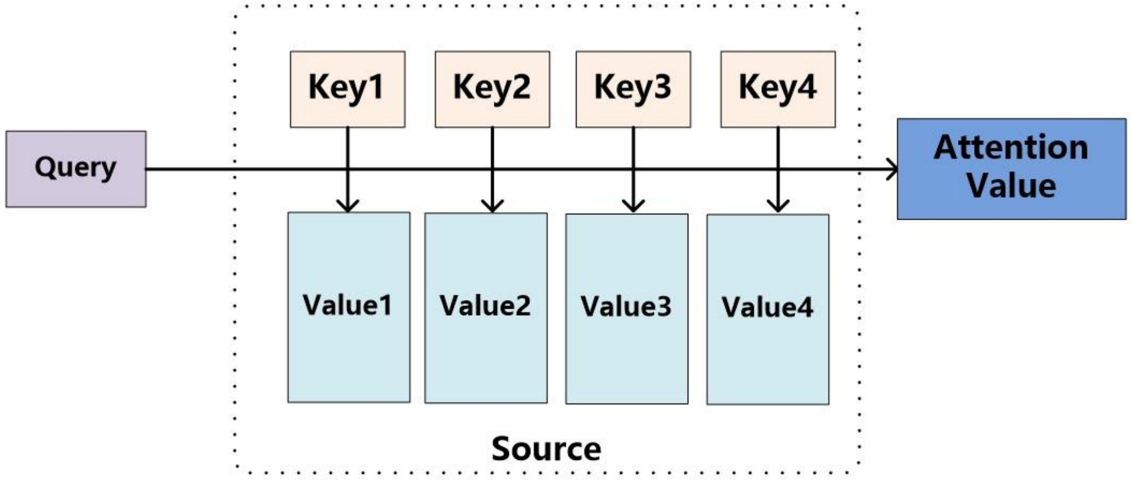
Encoder的流程如上图所示，最终的输出结果是每个时刻的hidden_state $h_1, h_2, h_3, \ldots, h_T$。

Seq2Seq Attention计算过程



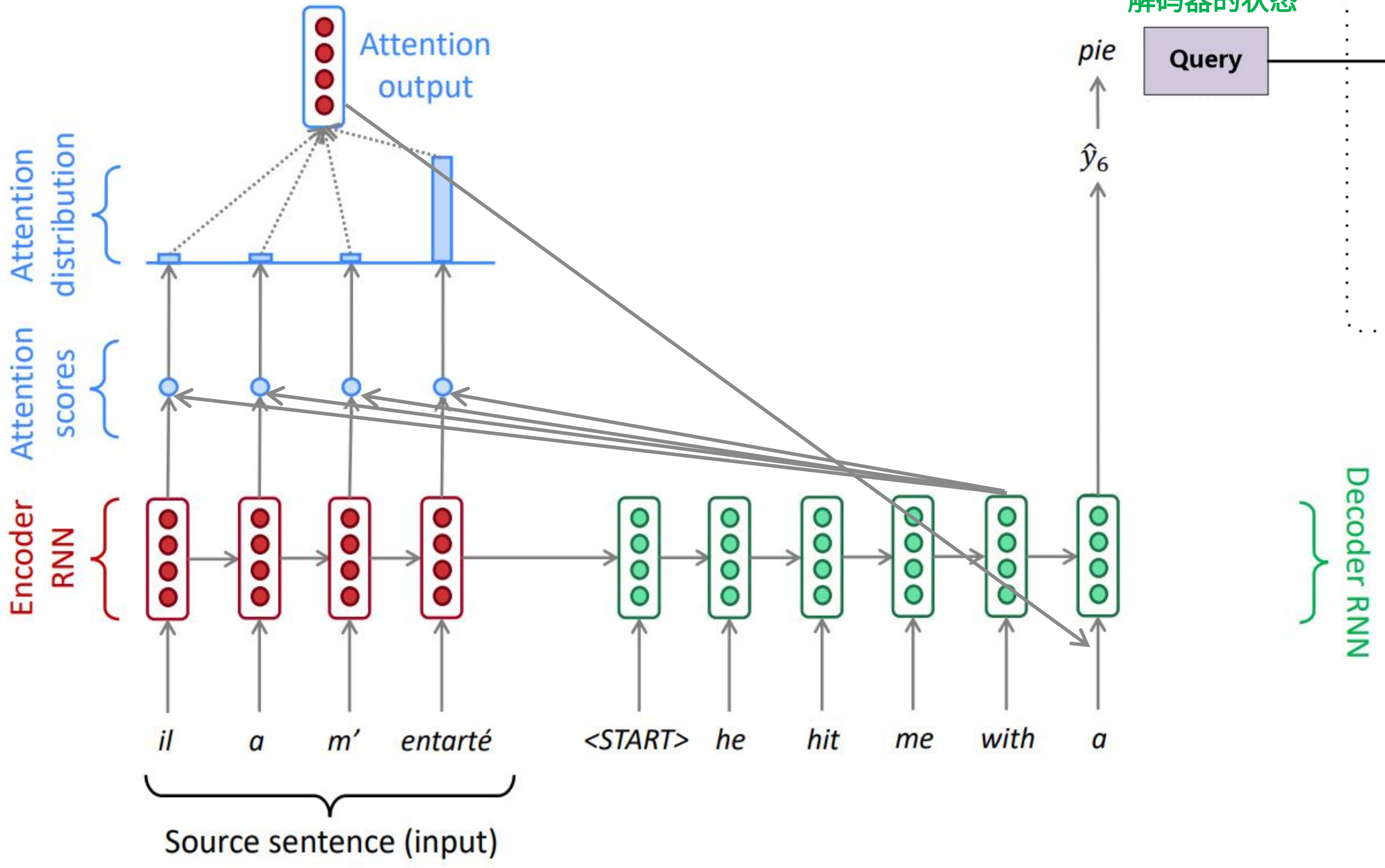
Seq2Seq Attention计算过程(另一种理解方式)

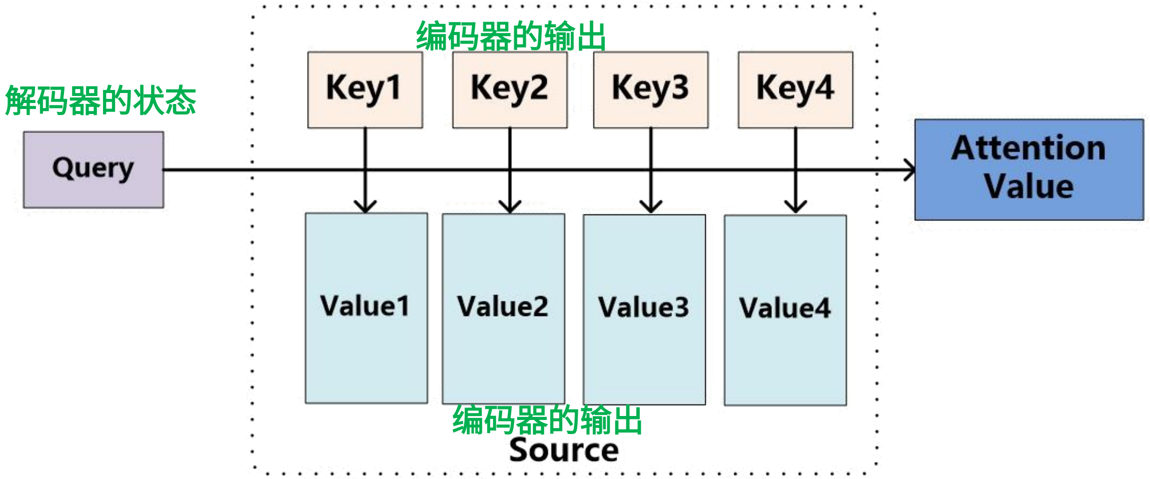
- 此时给定Target中的某个元素Query，通过计算Query和各个Key的相似性或者相关性，得到每个Key对应Value的权重系数，然后对Value进行加权求和，即得到了最终的Attention数值。所以**本质上Attention机制是对Source中元素的Value值进行加权求和，而Query和Key用来计算对应Value的权重系数。**



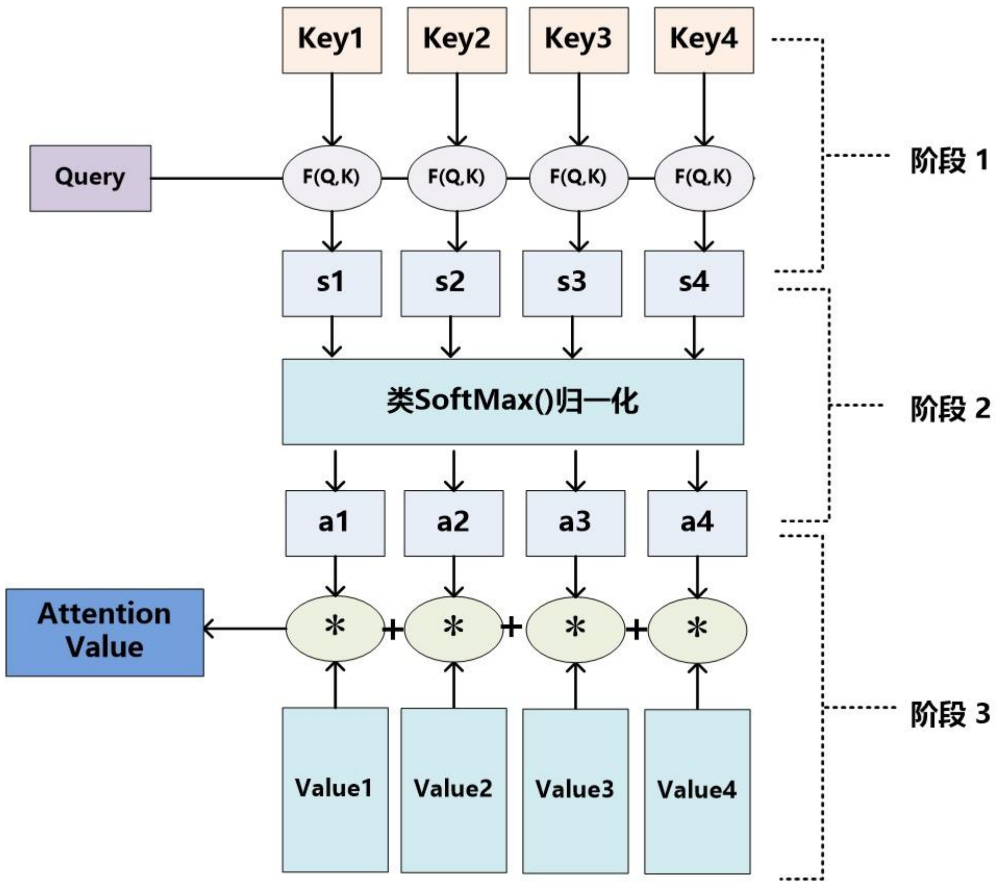
$$Attention(Query, Source) = \sum_{i=1}^{L_x} Similarity(Query, Key_i) * Value_i$$

Seq2Seq Attention计算过程(另一种理解方式)





Seq2Seq Attention计算过程(另一种理解方式)



点积： $\text{Similarity}(\text{Query}, Key_i) = \text{Query} \cdot Key_i$

Cosine 相似性： $\text{Similarity}(\text{Query}, Key_i) = \dfrac{\text{Query} \cdot Key_i}{||Query|| \cdot ||Key_i||}$

MLP 网络： $\text{Similarity}(\text{Query}, Key_i) = \text{MLP}(\text{Query}, Key_i)$

Seq2Seq Attention计算过程(另另一种理解方式)

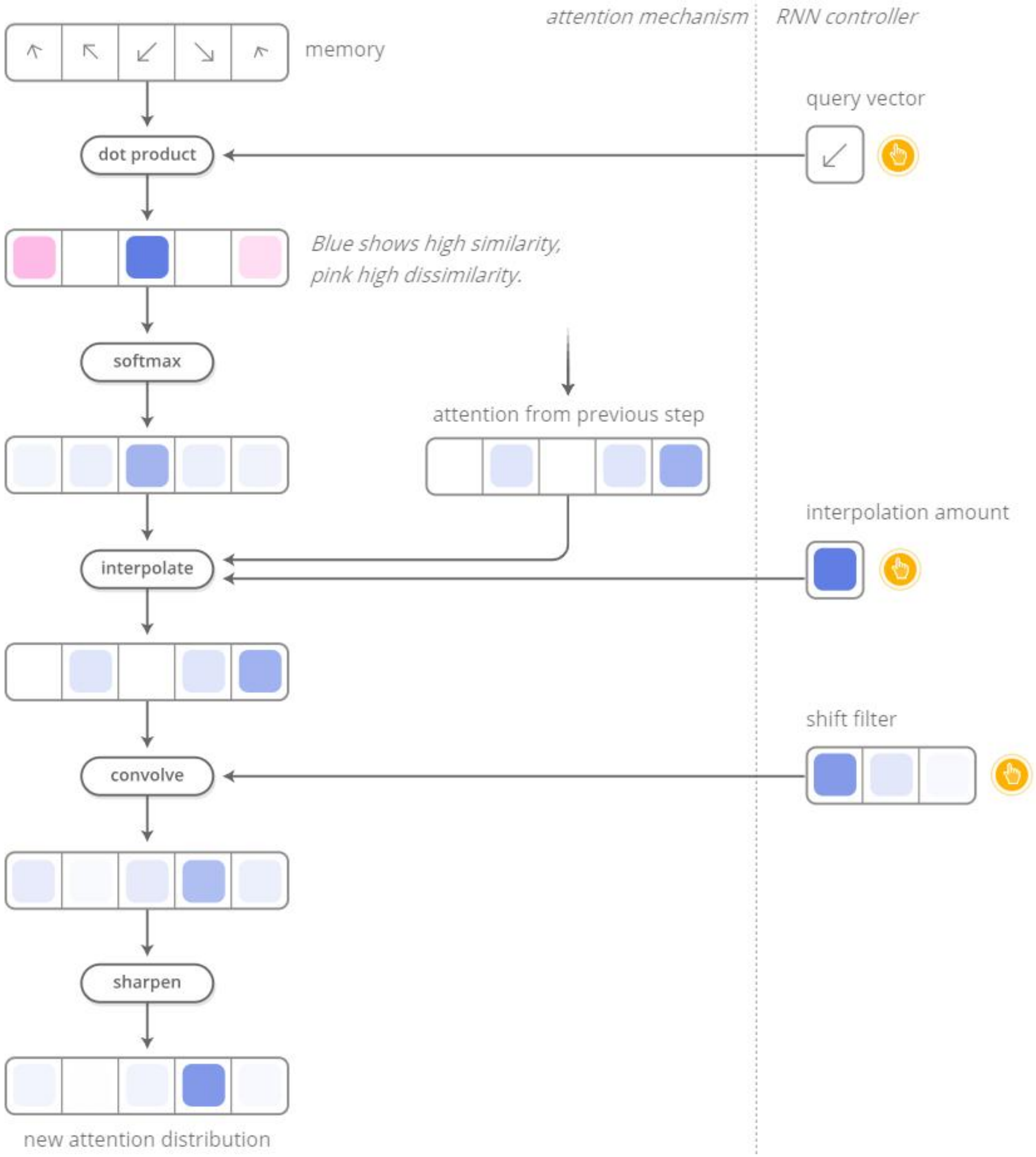
First, the controller gives a query vector and each memory entry is scored for similarity with the query.

The scores are then converted into a distribution using softmax.

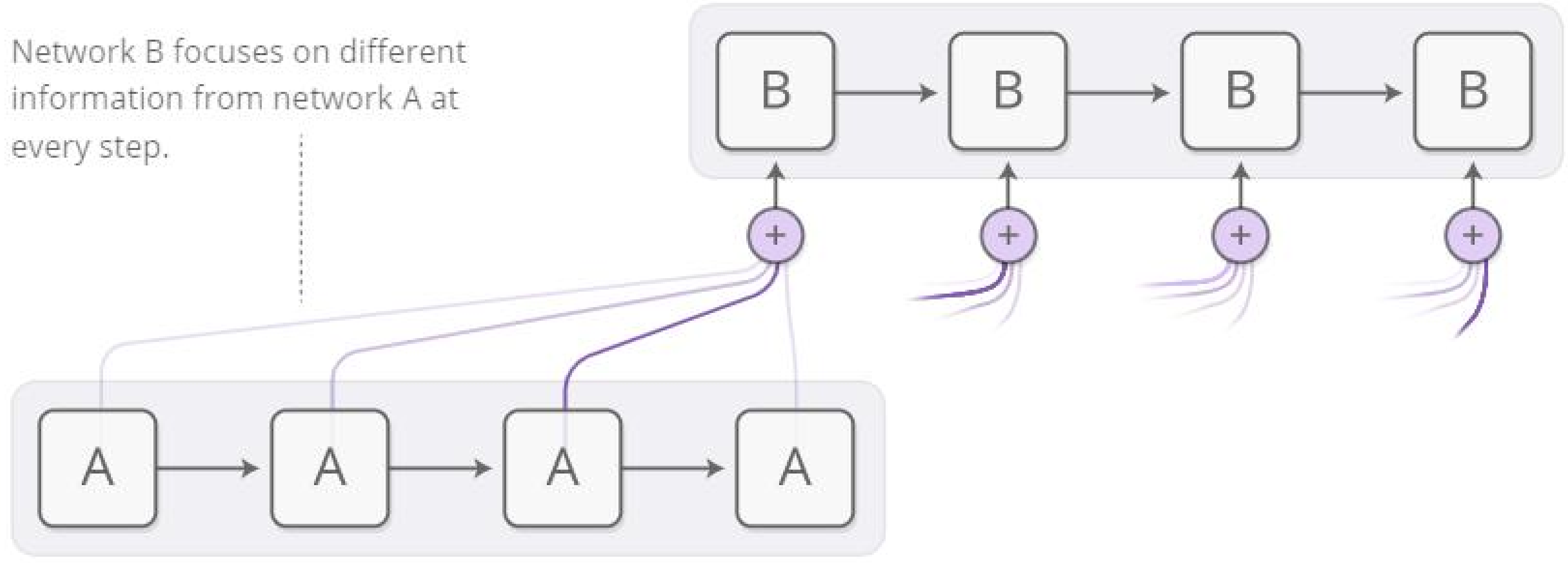
Next, we interpolate the attention from the previous time step.

We convolve the attention with a shift filter—this allows the controller to move its focus.

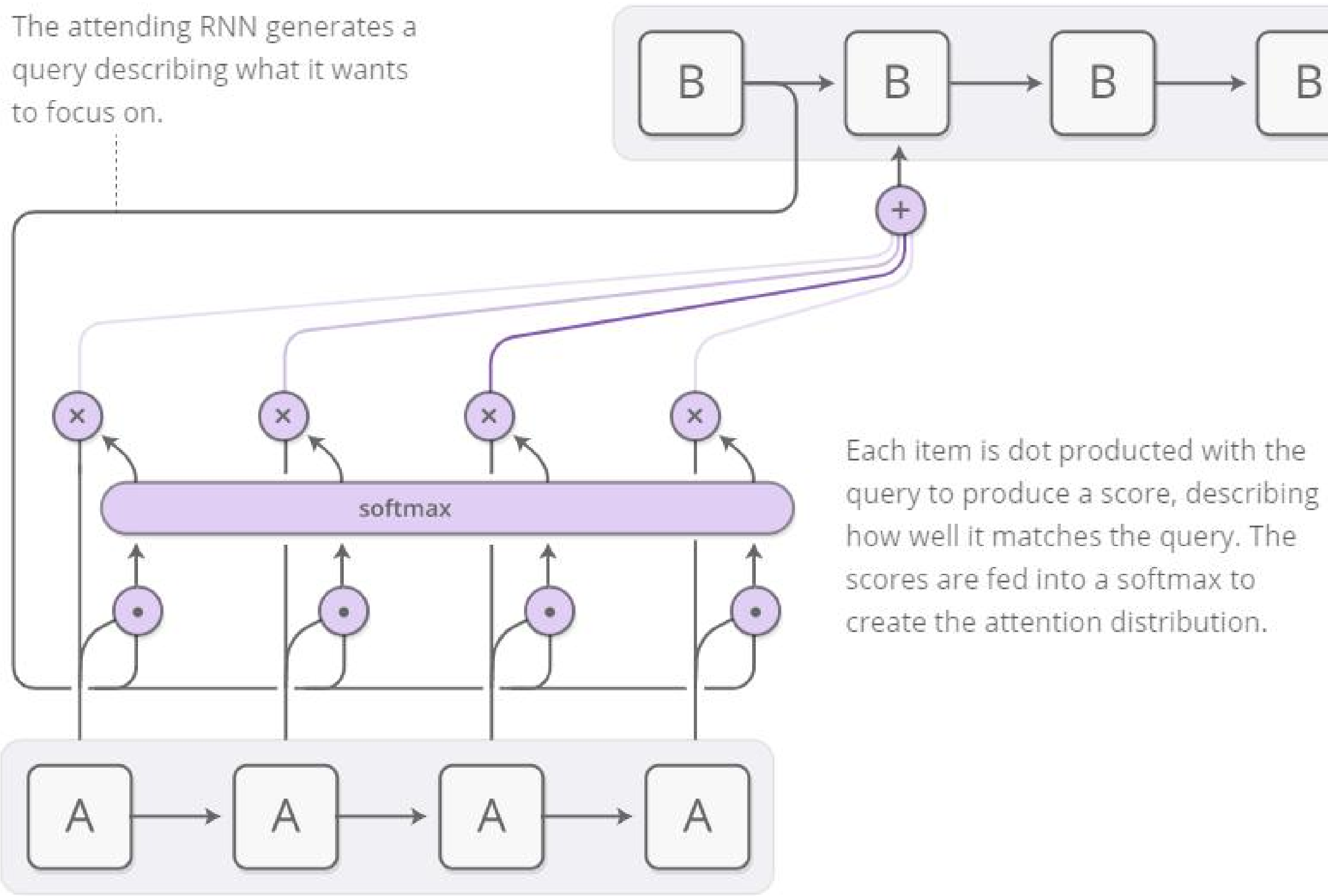
Finally, we sharpen the attention distribution. This final attention distribution is fed to the read or write operation.



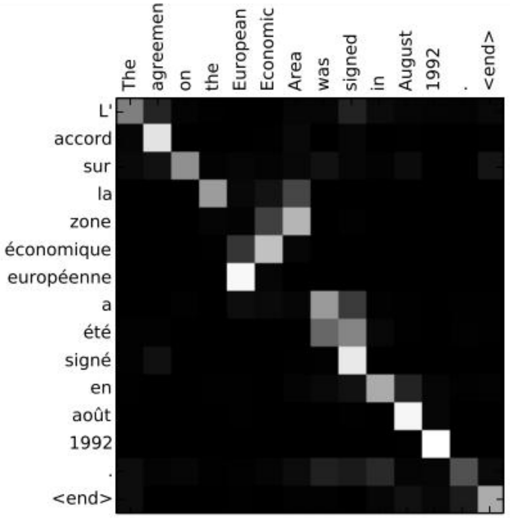
Seq2Seq Attention计算过程(另另一种理解方式)



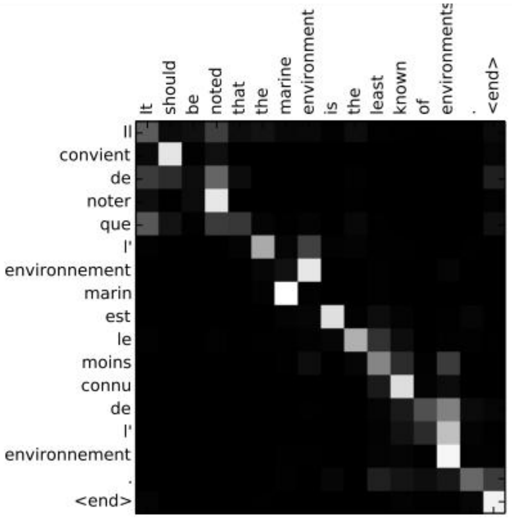
Seq2Seq Attention计算过程(另另一种理解方式)



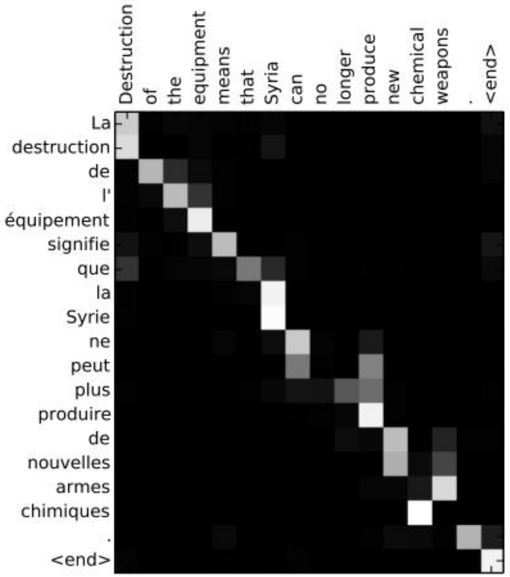
Seq2Seq Attention效果



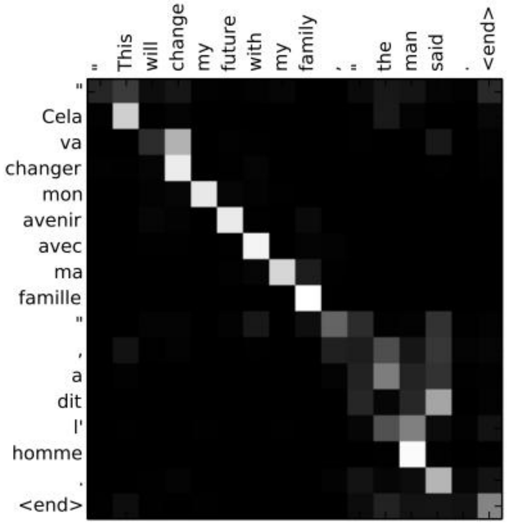
(a)



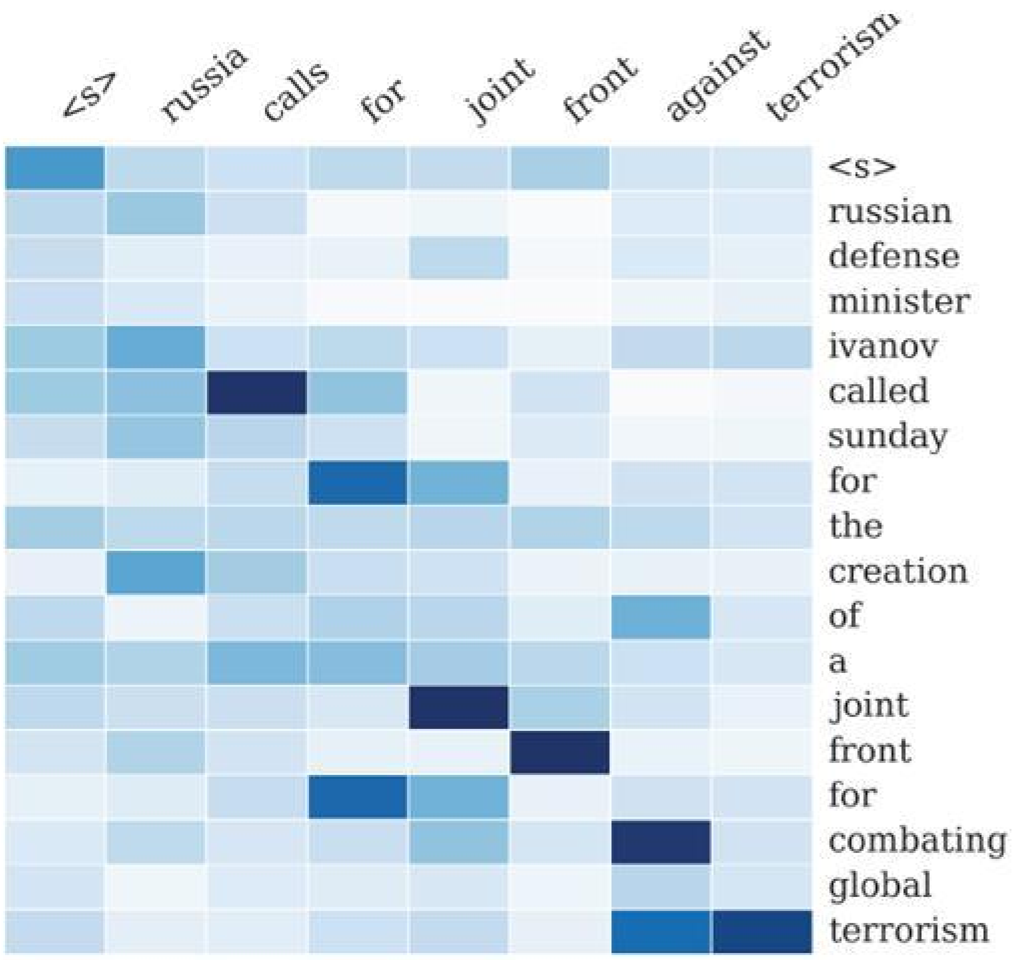
(b)



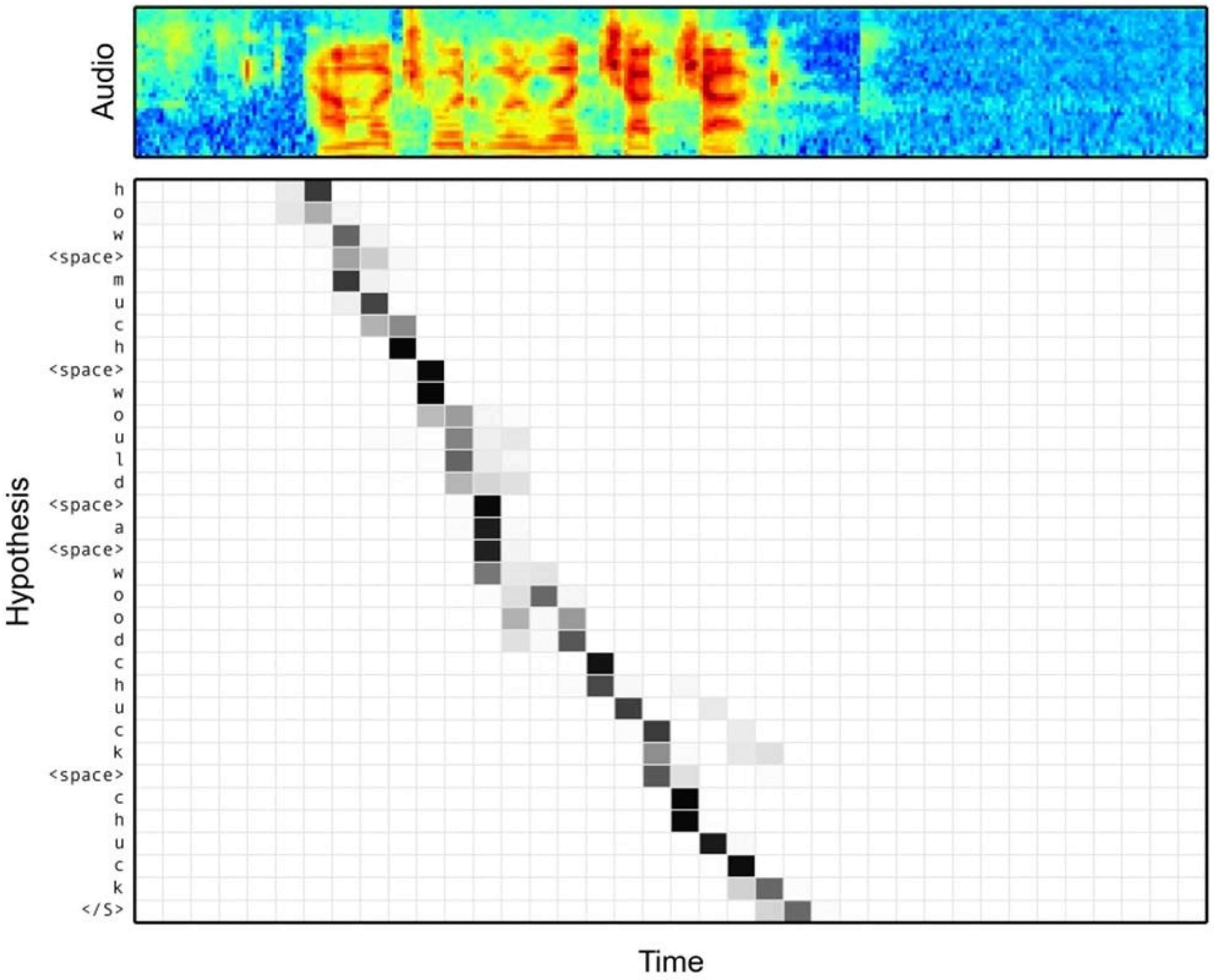
(c)



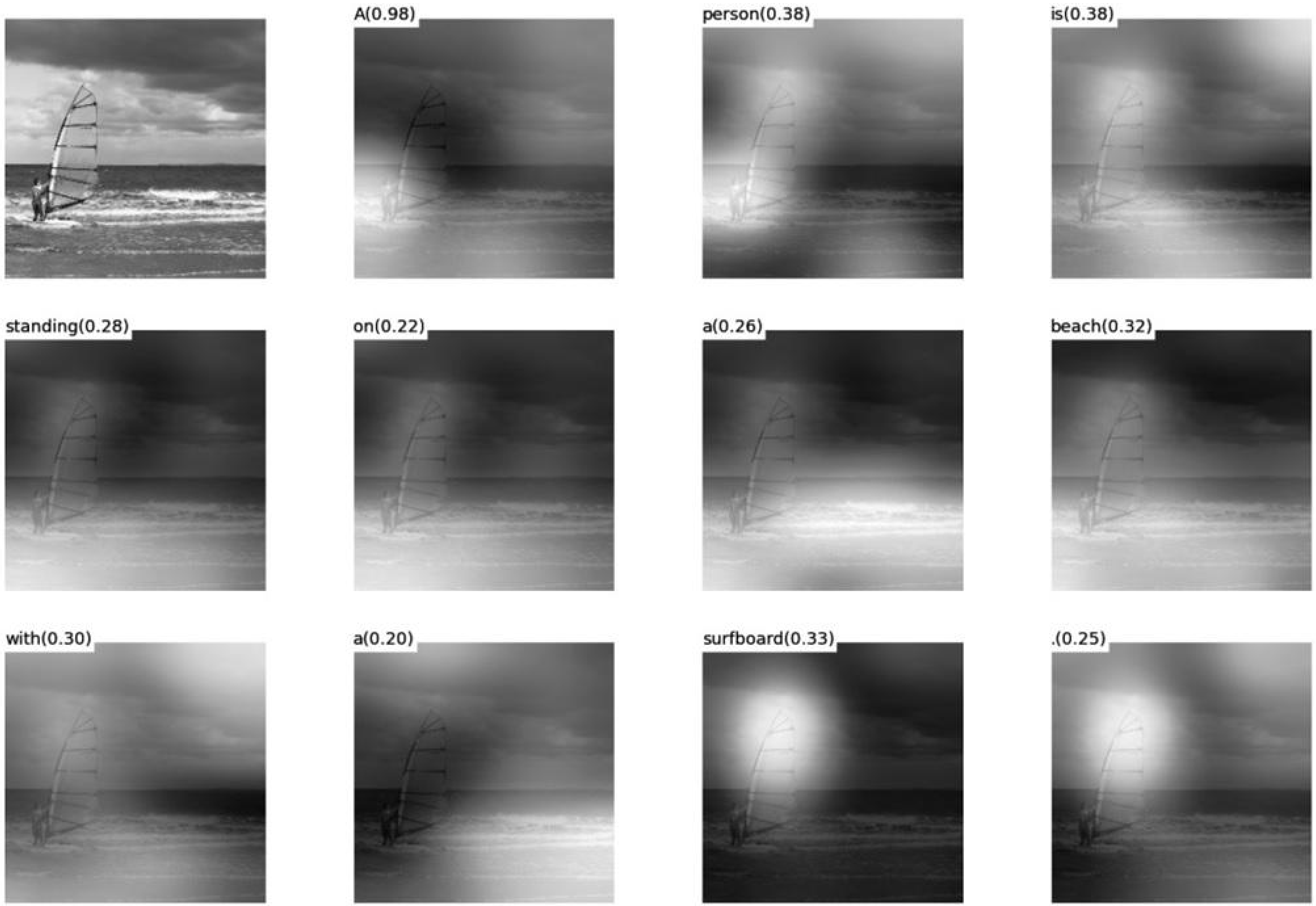
(d)



Seq2Seq Attention效果



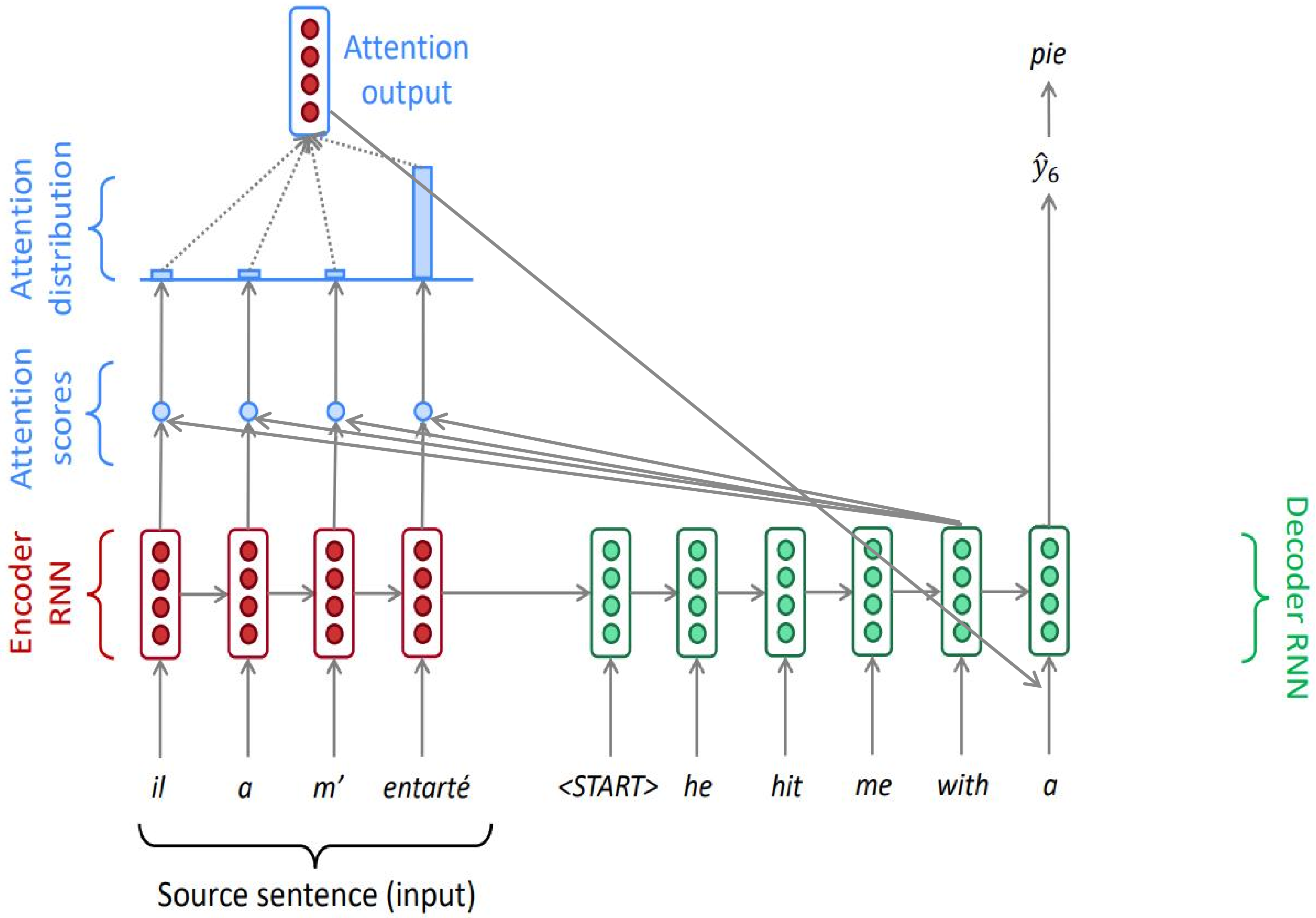
Seq2Seq Attention效果



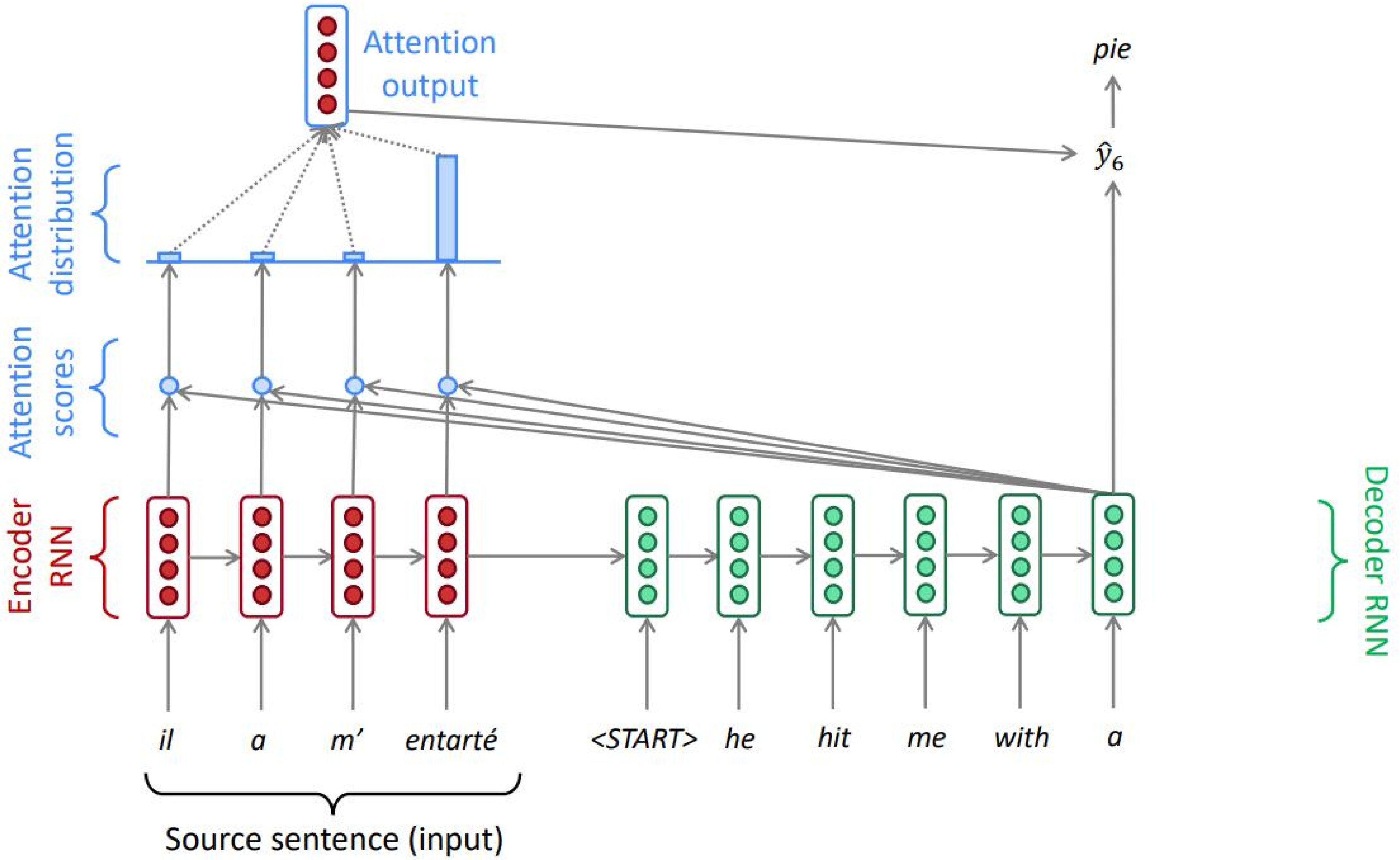
A woman is throwing a frisbee in a park.

A little girl sitting on a bed with a teddy bear.

Seq2Seq Attention常规形状一

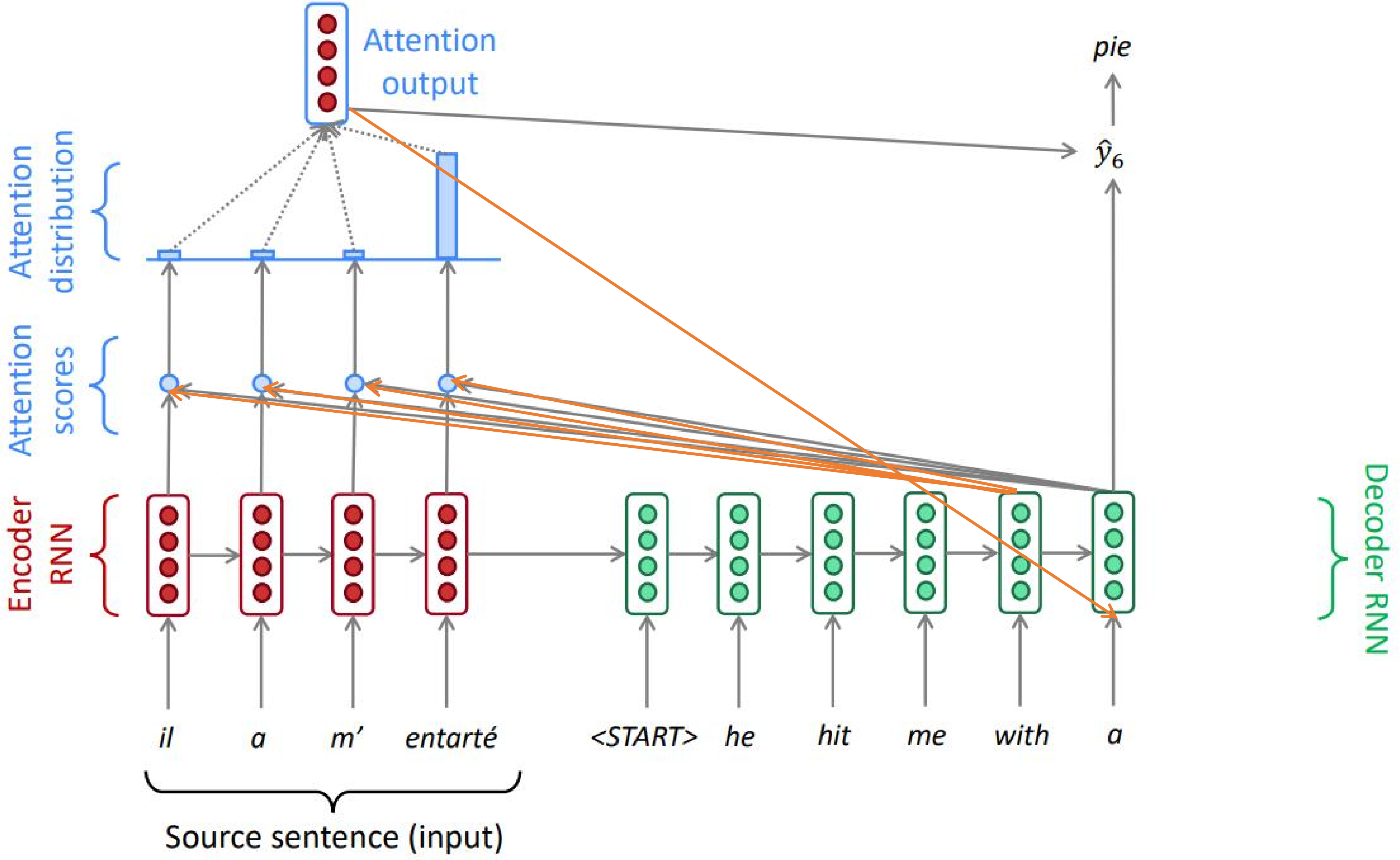


Seq2Seq Attention常规形状二



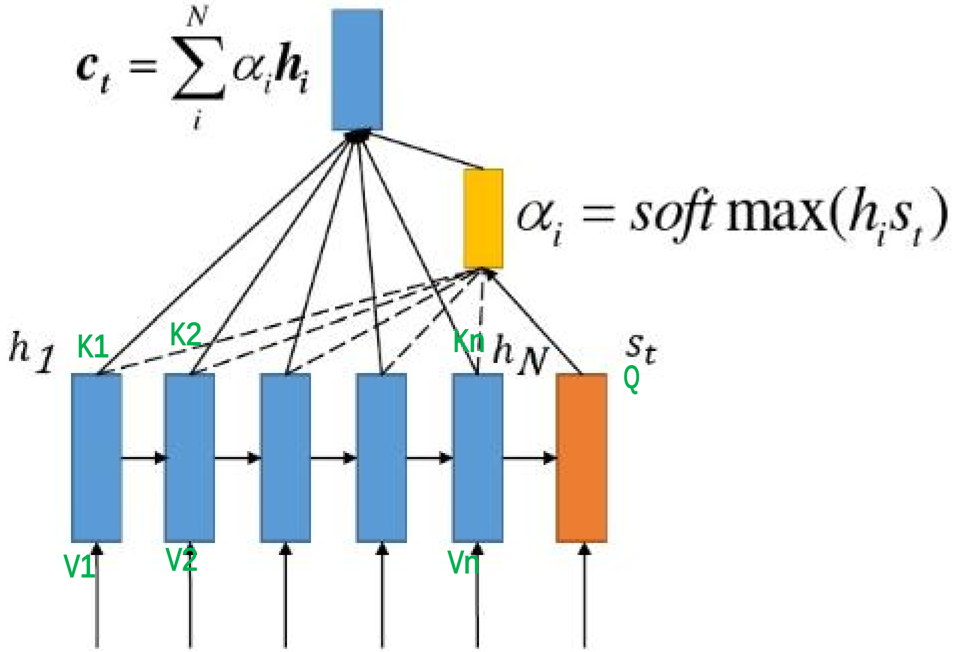
Seq2Seq Attention常规形状三

TensorFlow内部实现结构



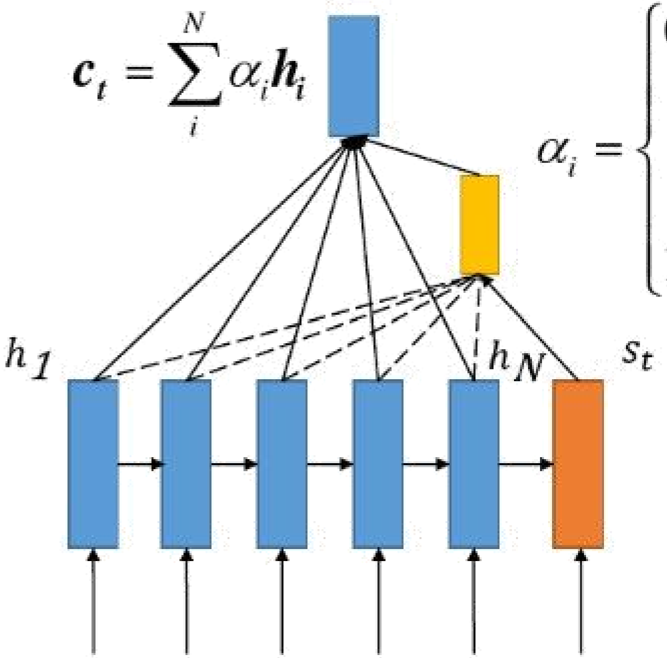
Seq2Seq Attention Soft Attention

- 15年被提出于《Show, Attend and Tell： Neural Image Caption Generation with Visual Attention, Kelvin Xu》



Seq2Seq Attention Hard Attention

- 和Soft Attention在15年同时在同一篇论文中被提出；
- Soft Attention中是对于每个Encoder的Hidden State会match一个概率值，而在Hard Attention会直接找一个特定的单词概率为1，而其它对应概率为0.



$$\alpha_i = \begin{cases} 0 & \text{else} \\ 1 & \text{if i==pt} \end{cases}$$

Seq2Seq Attention Global Attention

- 在15年被提出于《Effective Approaches to Attention-based Neural Machine Translation, Minh-Thang Luong》，和Soft Attention类似。

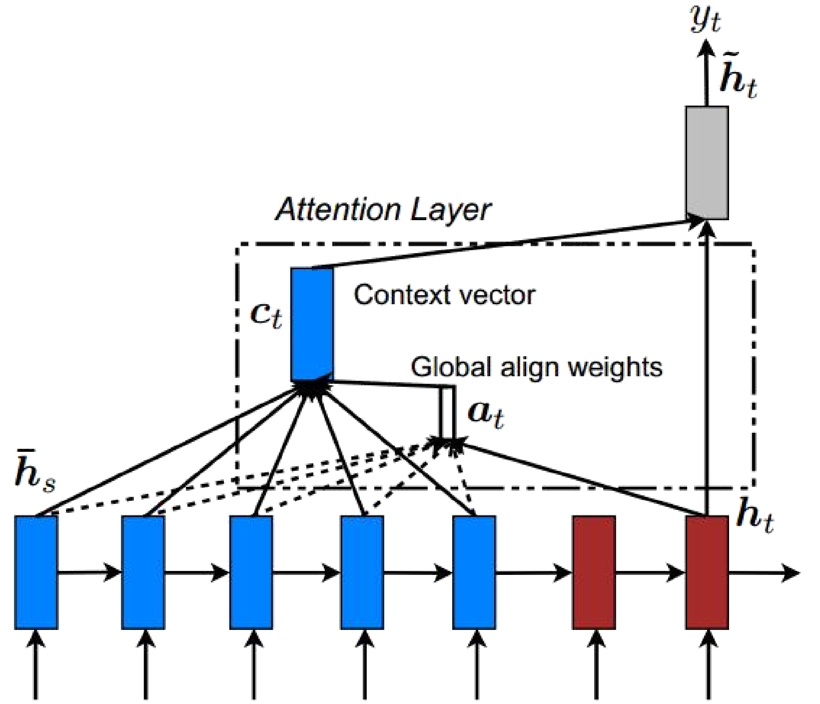


Figure 2: **Global attentional model** – at each time step t, the model infers a *variable-length* alignment weight vector $\boldsymbol{a}_t$ based on the current target state $\boldsymbol{h}_t$ and all source states $\bar{\boldsymbol{h}}_s$. A global context vector $\boldsymbol{c}_t$ is then computed as the weighted average, according to $\boldsymbol{a}_t$, over all the source states.

$$\boldsymbol{h}_j = f(\boldsymbol{h}_{j-1}, \boldsymbol{s})$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t])$$

$$p(y_t | y_{<t}, x) = \mathrm{softmax}(\boldsymbol{W}_{\boldsymbol{s}} \tilde{\boldsymbol{h}}_t)$$

$$\begin{aligned}\boldsymbol{a}_t(s) &= \mathrm{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)}\end{aligned}$$

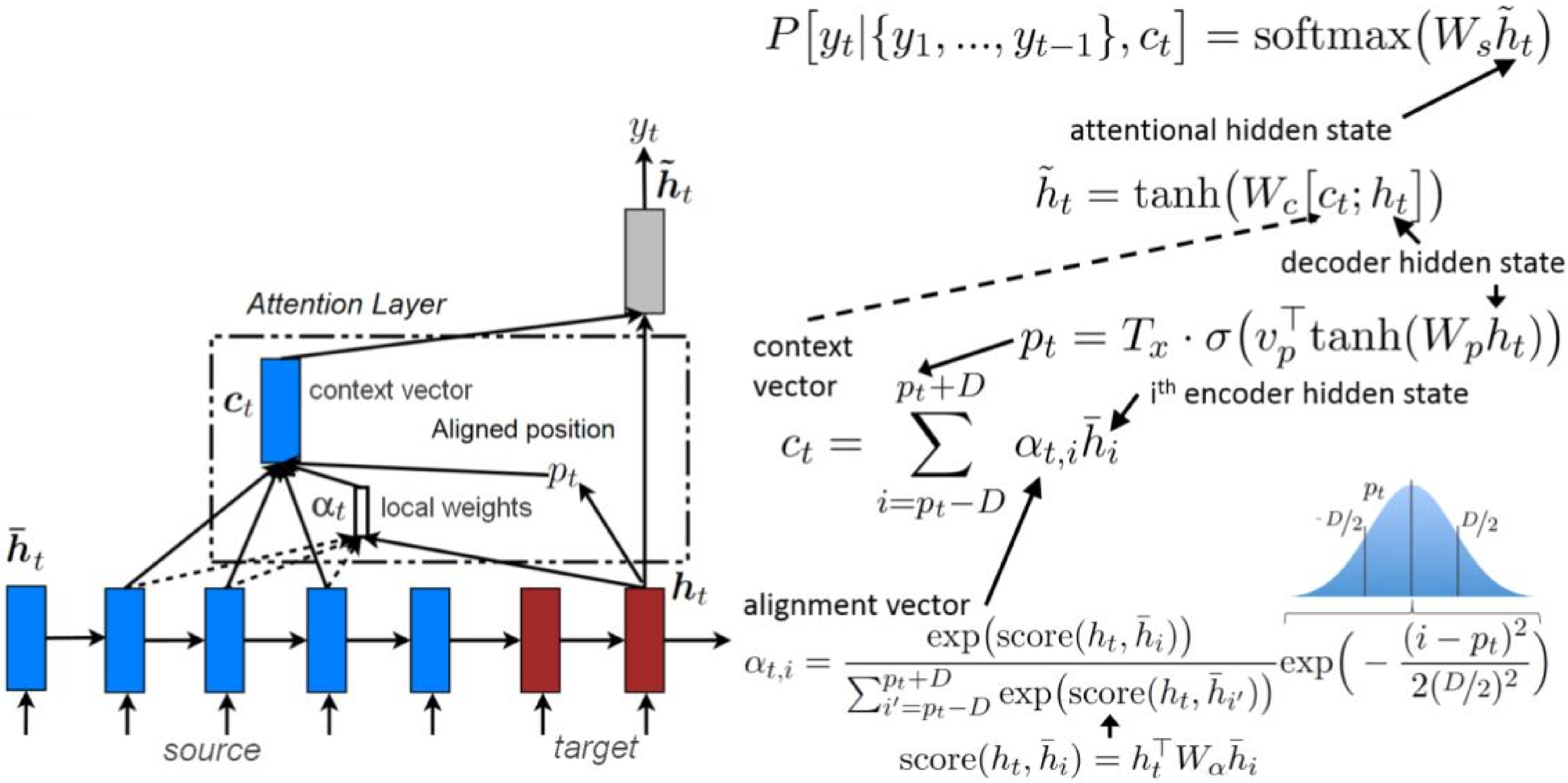
$$\mathrm{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_{\boldsymbol{a}} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\boldsymbol{a}_t = \mathrm{softmax}(\boldsymbol{W}_{\boldsymbol{a}} \boldsymbol{h}_t) \qquad \textit{location}$$

Seq2Seq Attention Local Attention

- 和Global Attention在同一篇论文中被提出；相当于Soft Attention和Hard Attention中间状态(半硬半软Attention)
- 对于时刻t的词汇，模型首先产生一个对齐位置pt(aligned position)，context vector(c)由编码器中的隐状态计算得到，编码器的隐状态不是所有的隐状态，而是在区间[pt-D, pt+D]中，D的大小由经验给定。

Seq2Seq Attention Local Attention



Seq2Seq Attention Self Attention

- 在17年被提出于《Attention Is All You Need, Ashish Vaswani》，也称为Transformer结构；内部包含Multi-Head Attention以及Rest残差结构。
- Transformer是Bert、LLM等网络结构的基础。

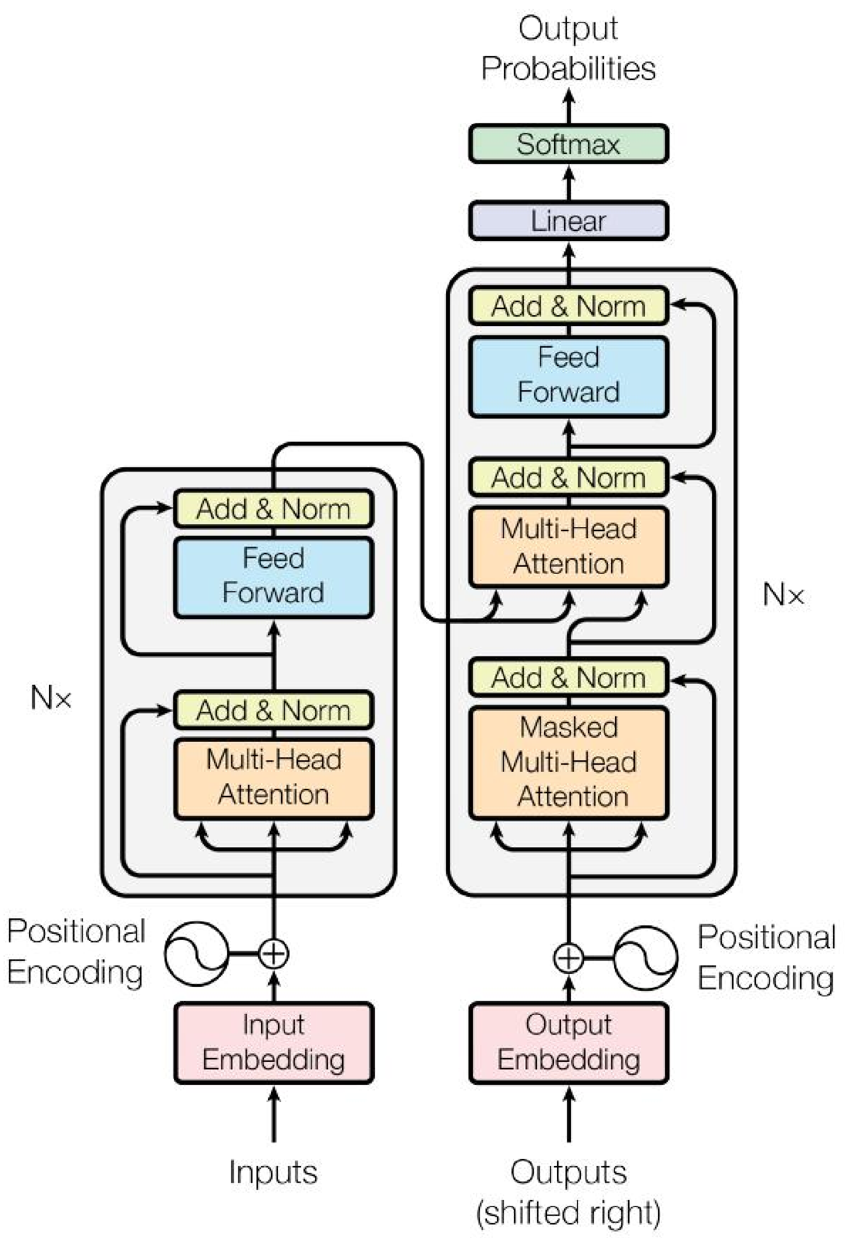
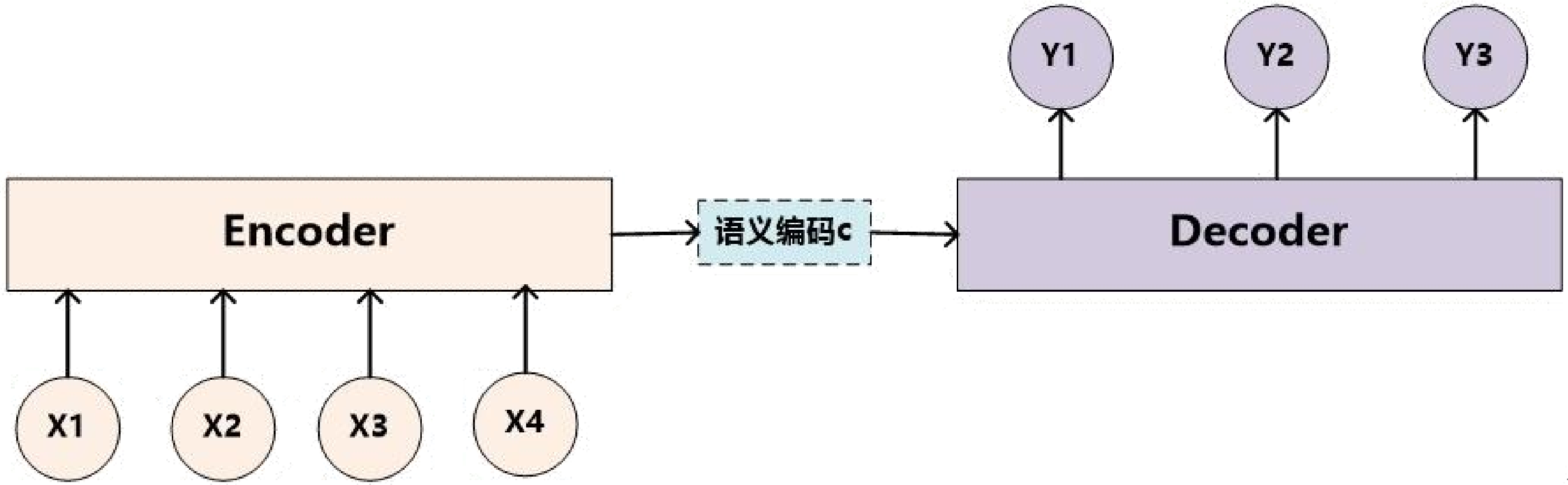


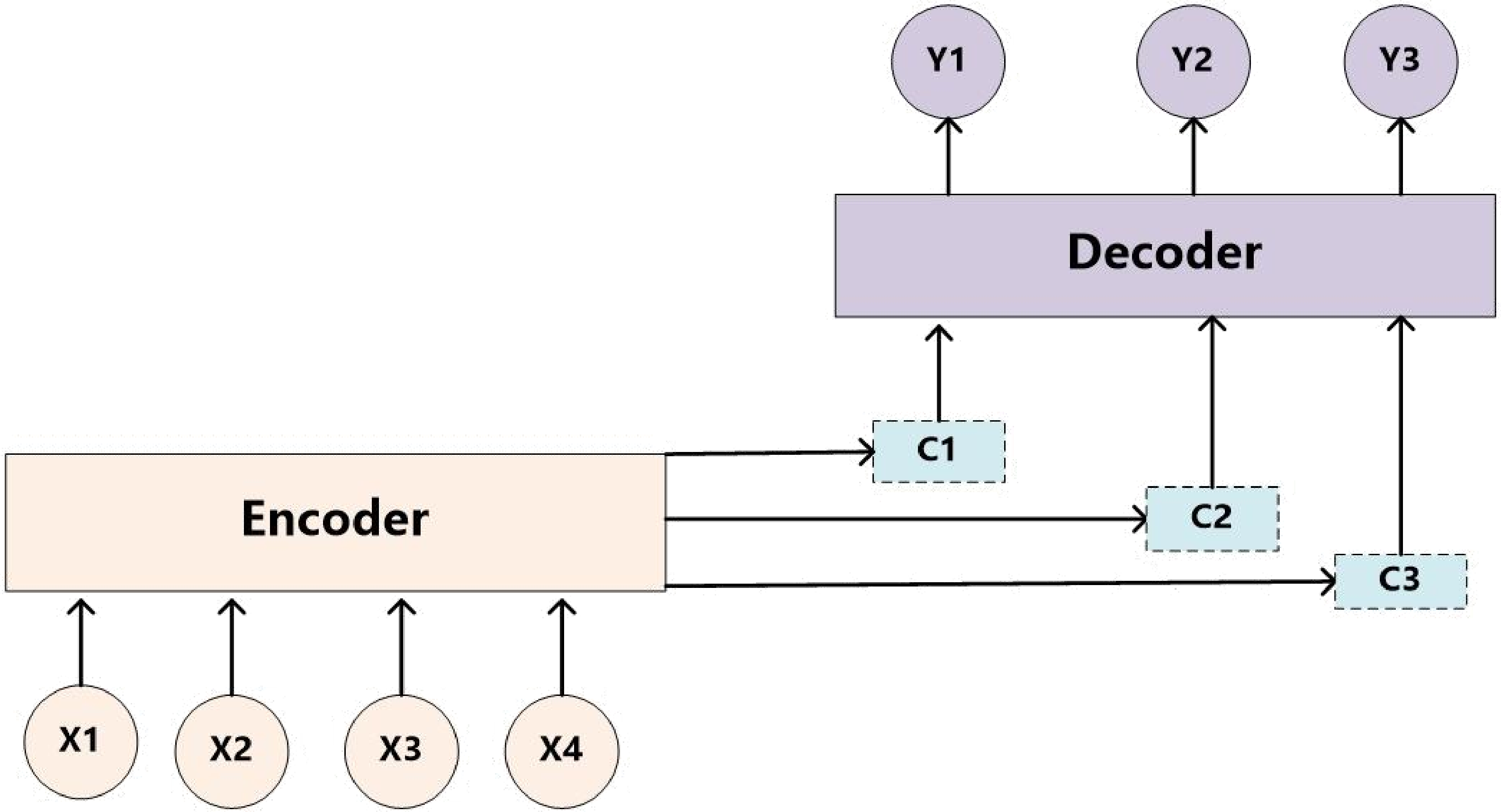
Figure 1: The Transformer - model architecture.

总结_Seq2Seq_Attention



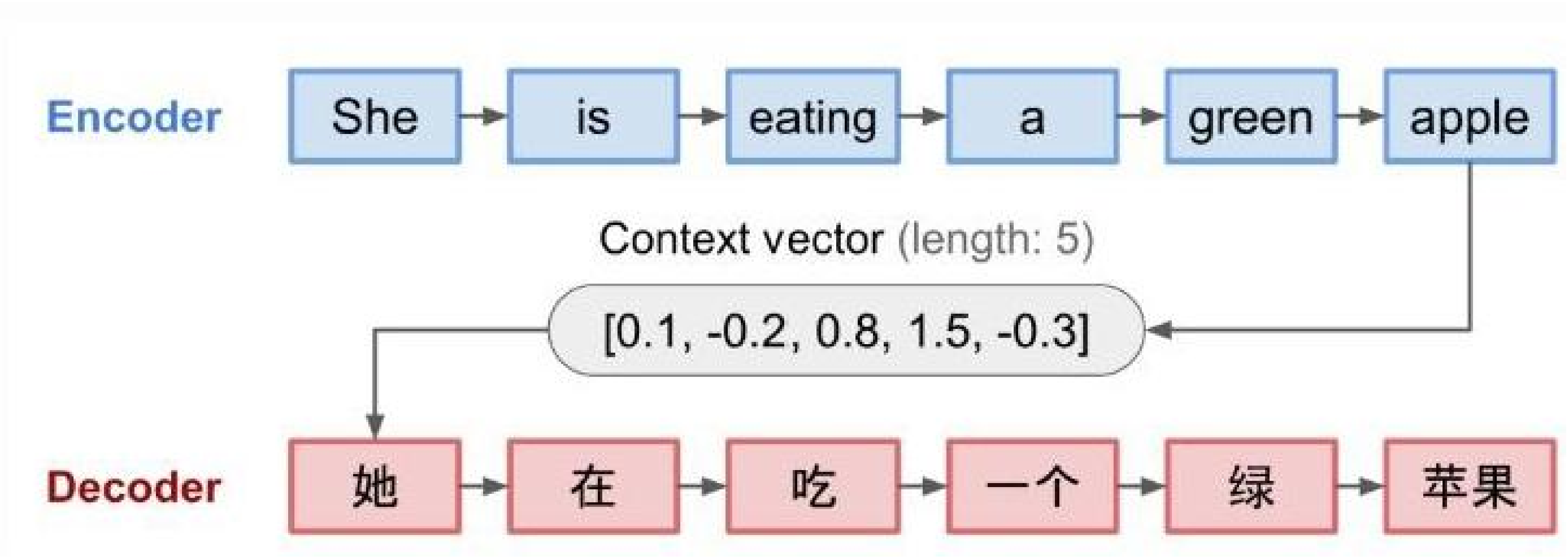
$$Y_1 = f(C)$$
$$Y_2 = f(C, Y_1)$$
$$Y_3 = f(C, Y_1, Y_2)$$

总结_Seq2Seq_Attention

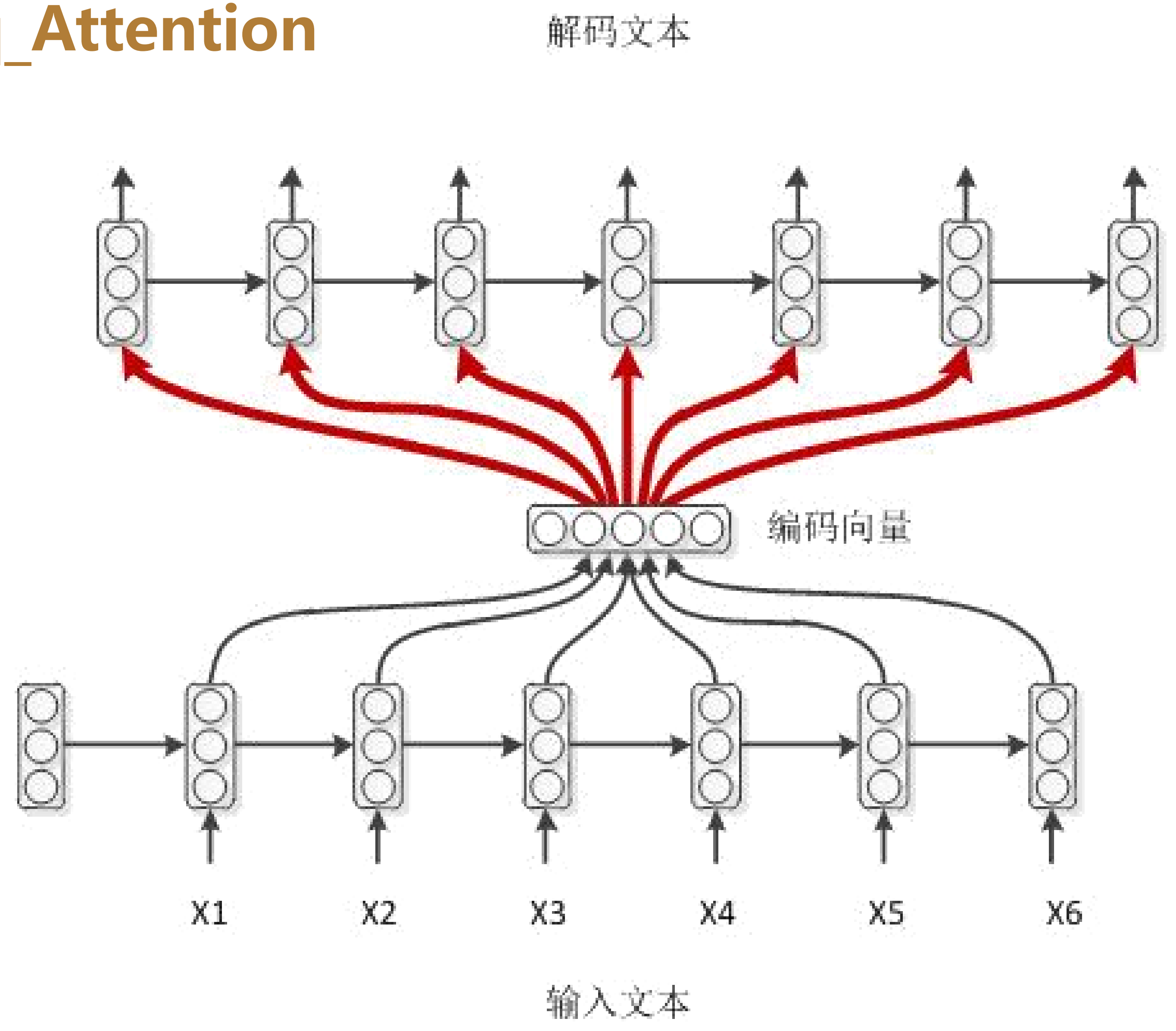


$$Y_1 = f(C_1)$$
$$Y_2 = f(C_2, Y_1)$$
$$Y_3 = f(C_3, Y_1, Y_2)$$

总结_Seq2Seq_Attention

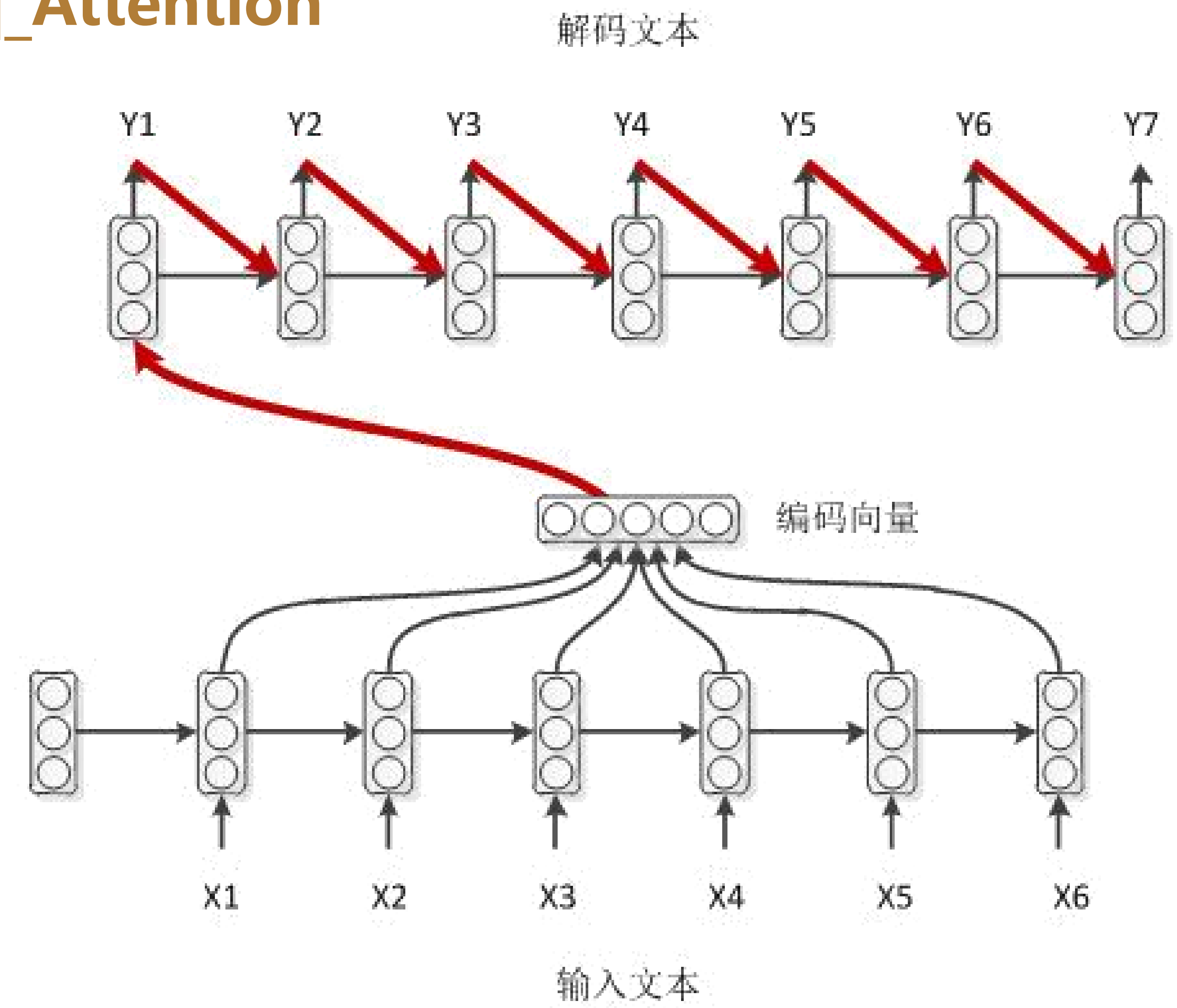


总结_Seq2Seq_Attention



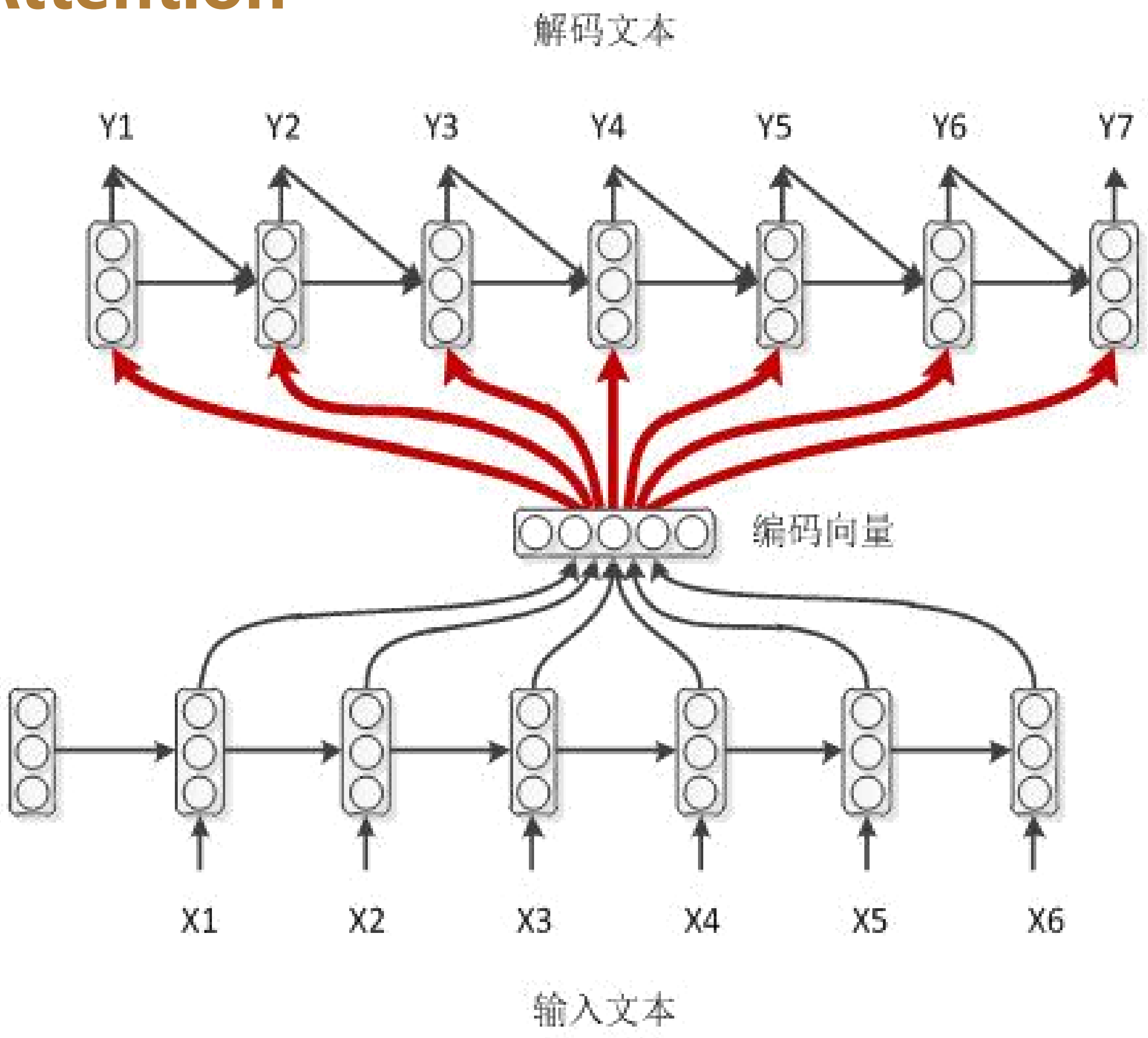
- **最简单的解码模式 – Decoder 1**

总结_Seq2Seq_Attention



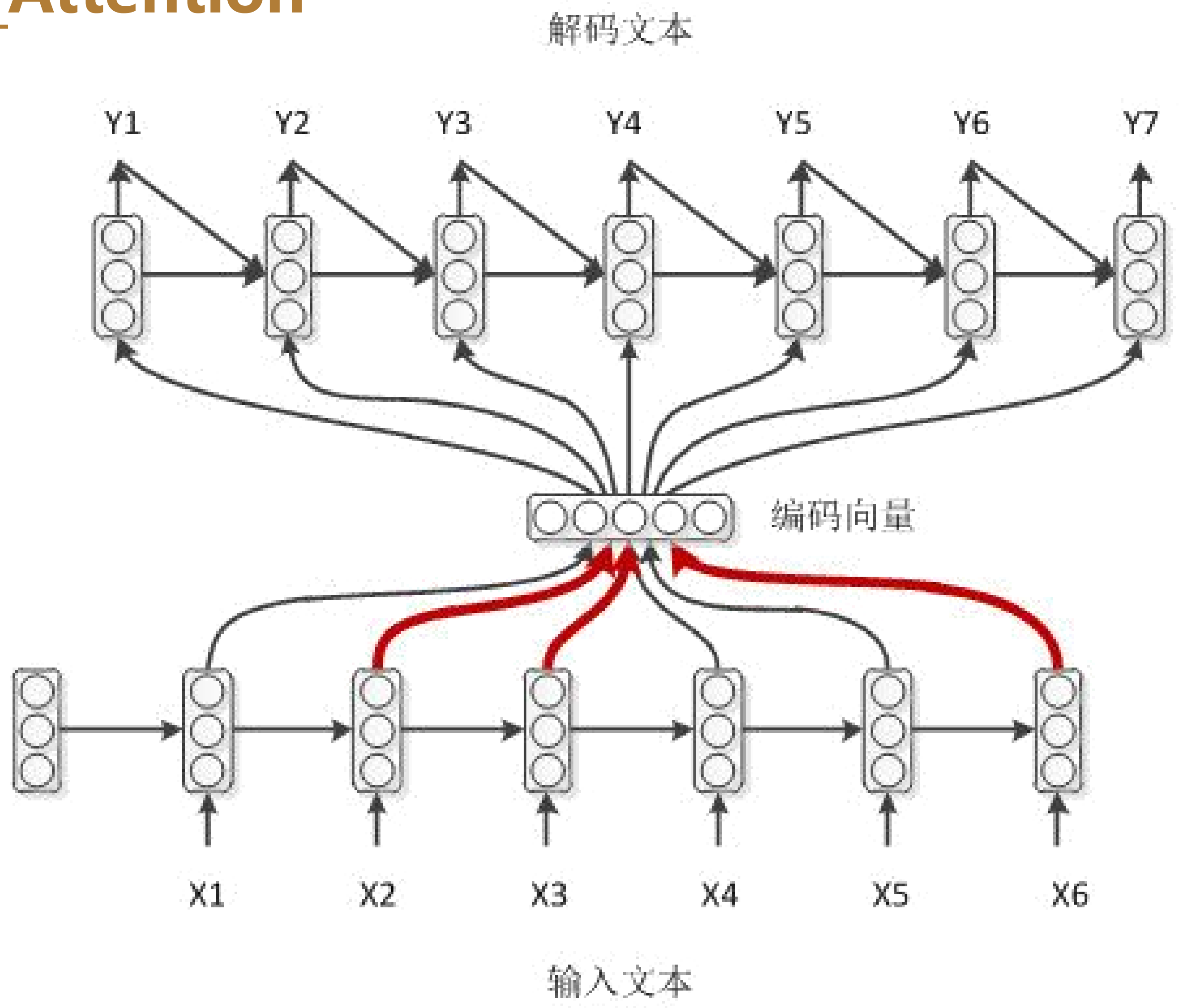
- **带输出回馈的解码模式 – Decoder 2**

总结_Seq2Seq_Attention



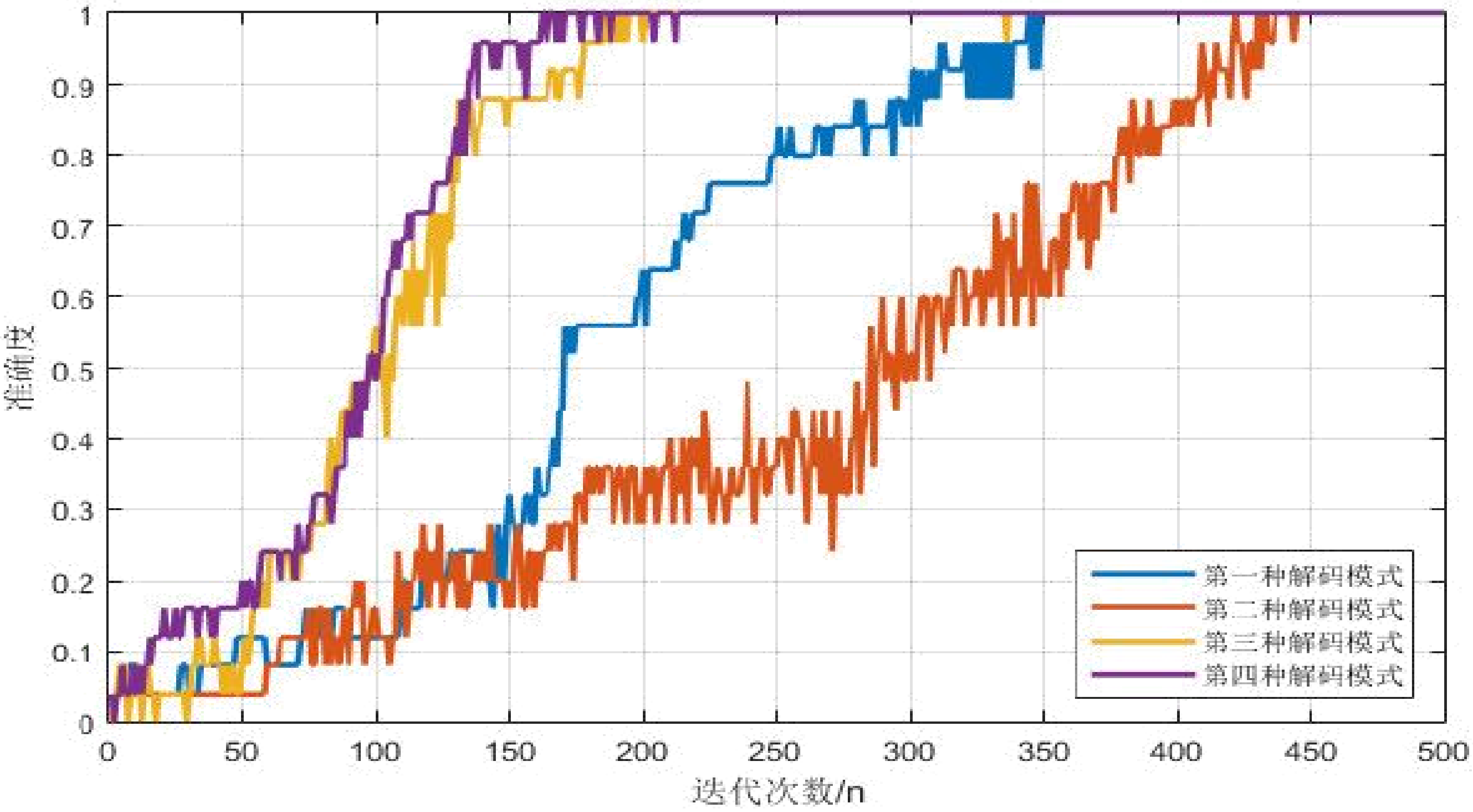
- **带编码向量的解码模式– Decoder 3**

总结_Seq2Seq_Attention

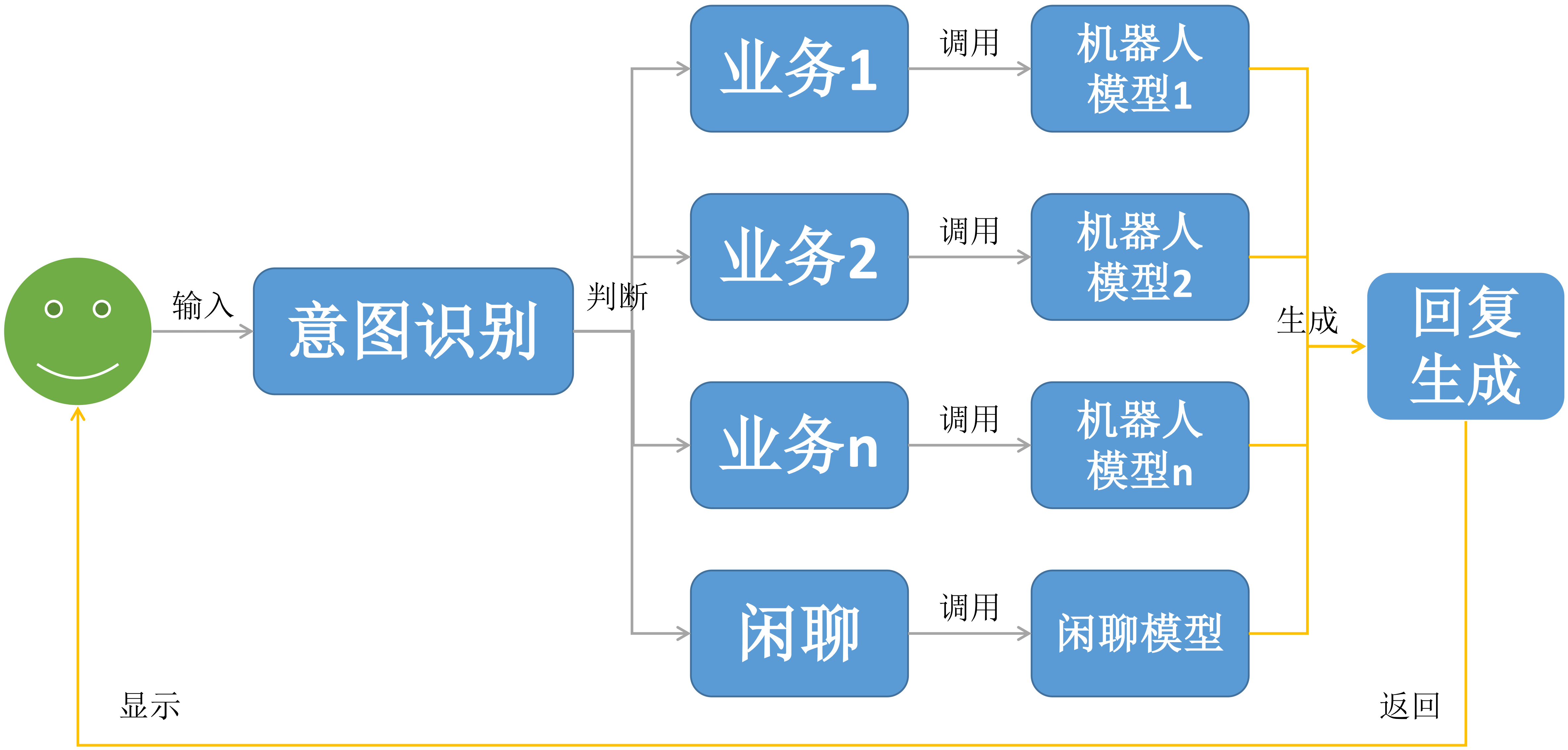


- **带注意力的解码模式– Decoder 4**

总结_Seq2Seq_Attention



Seq2Seq+Attention项目_聊天机器人



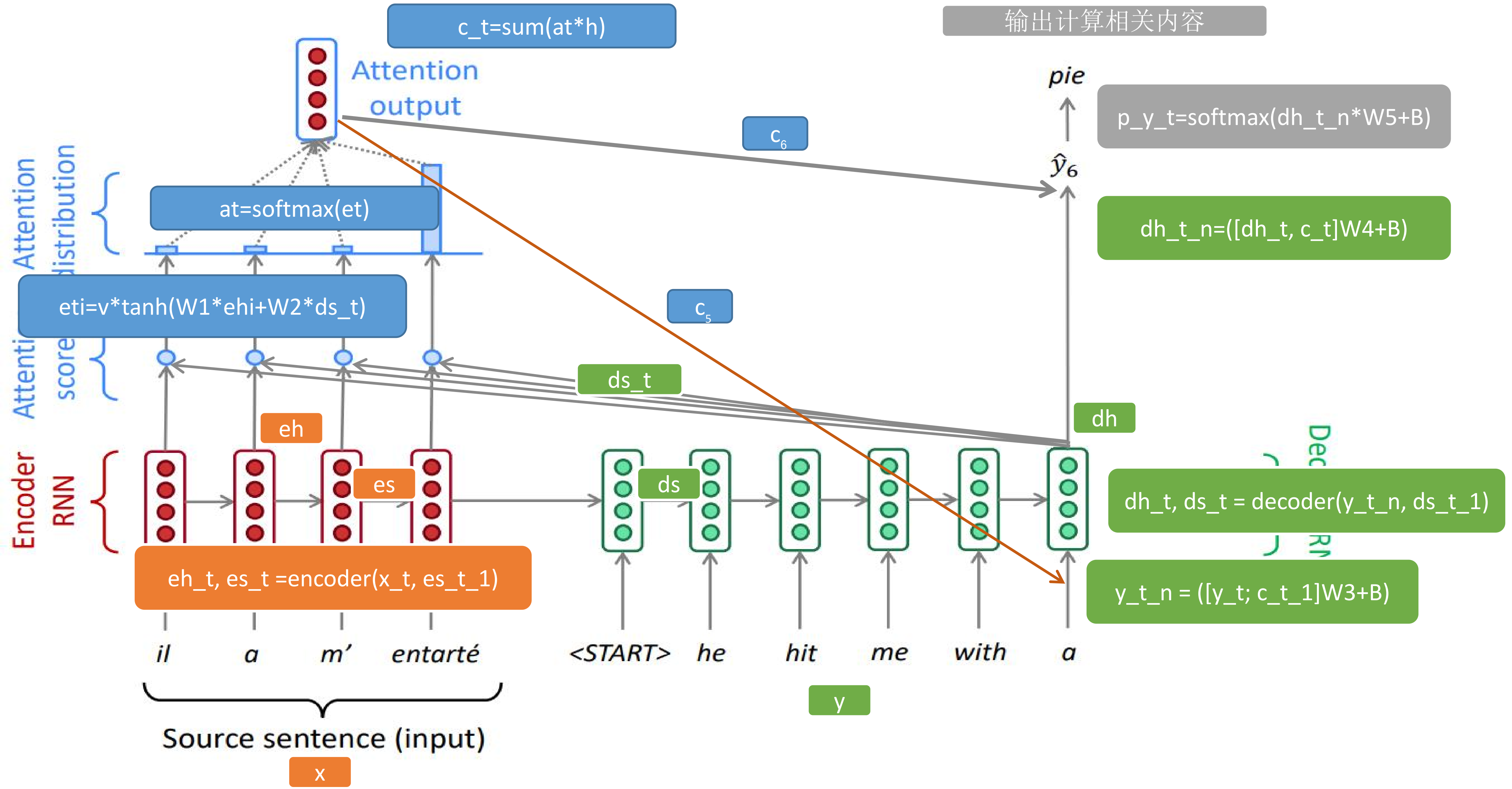
Seq2Seq Attention实现

编码器相关内容

解码器相关内容

Attention计算相关内容

输出计算相关内容



THANKS!