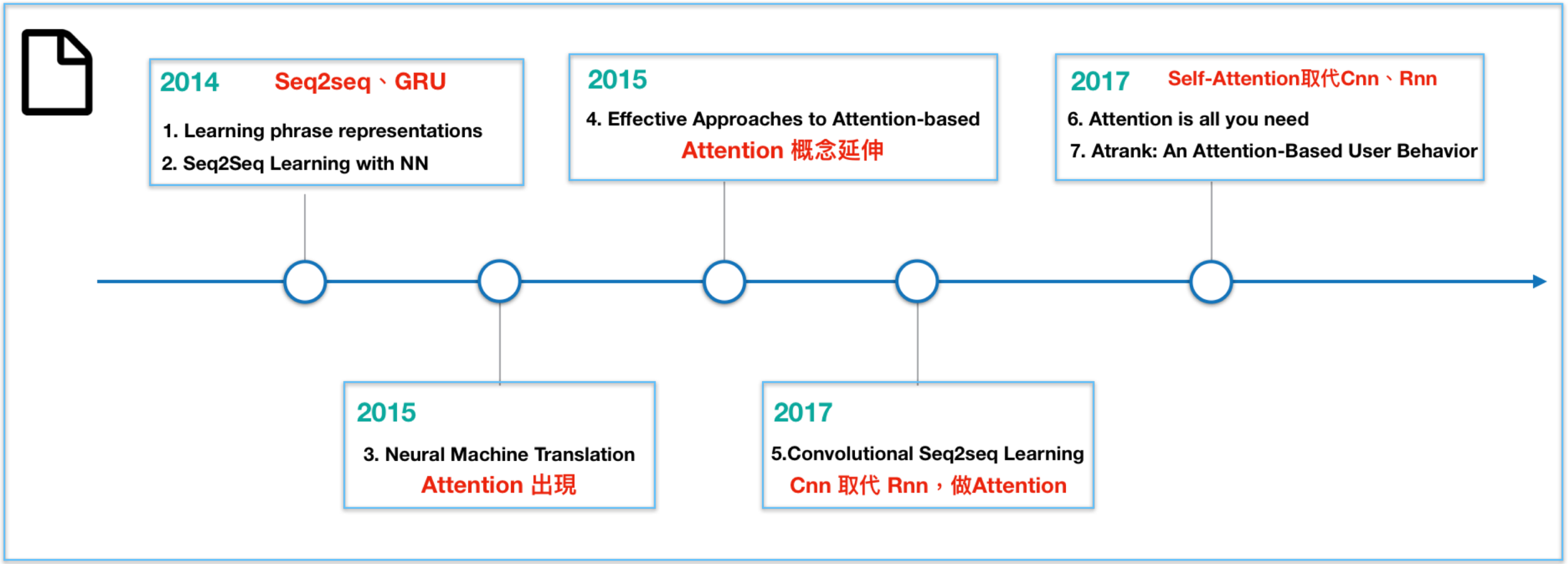
Attention Mechanism

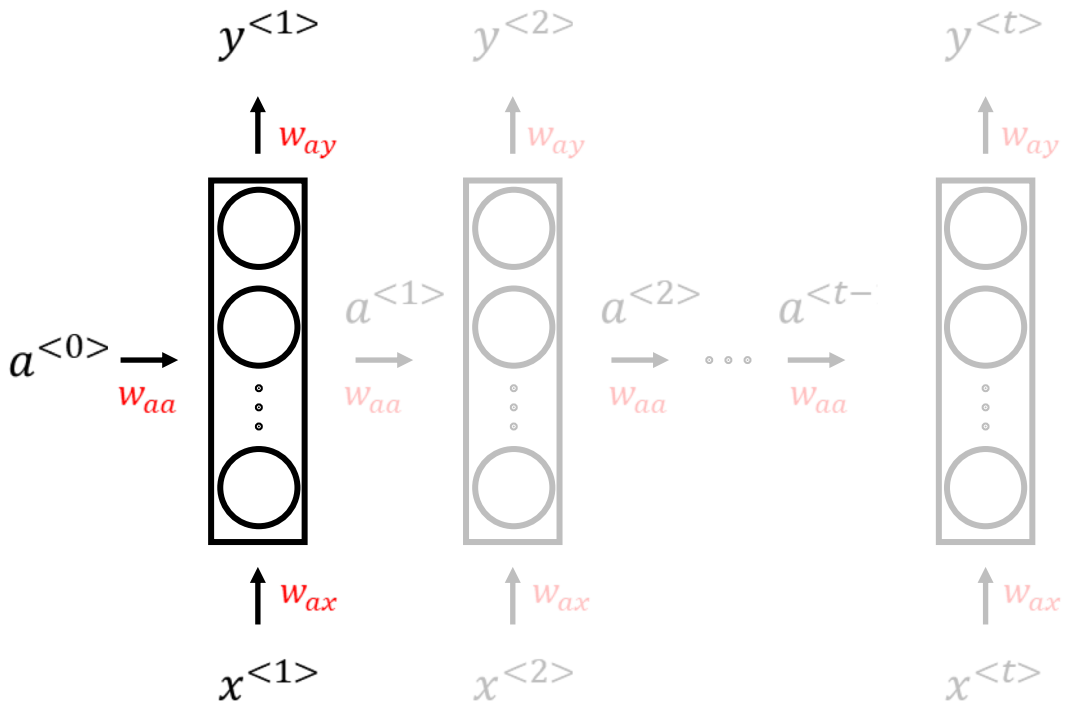
Agenda

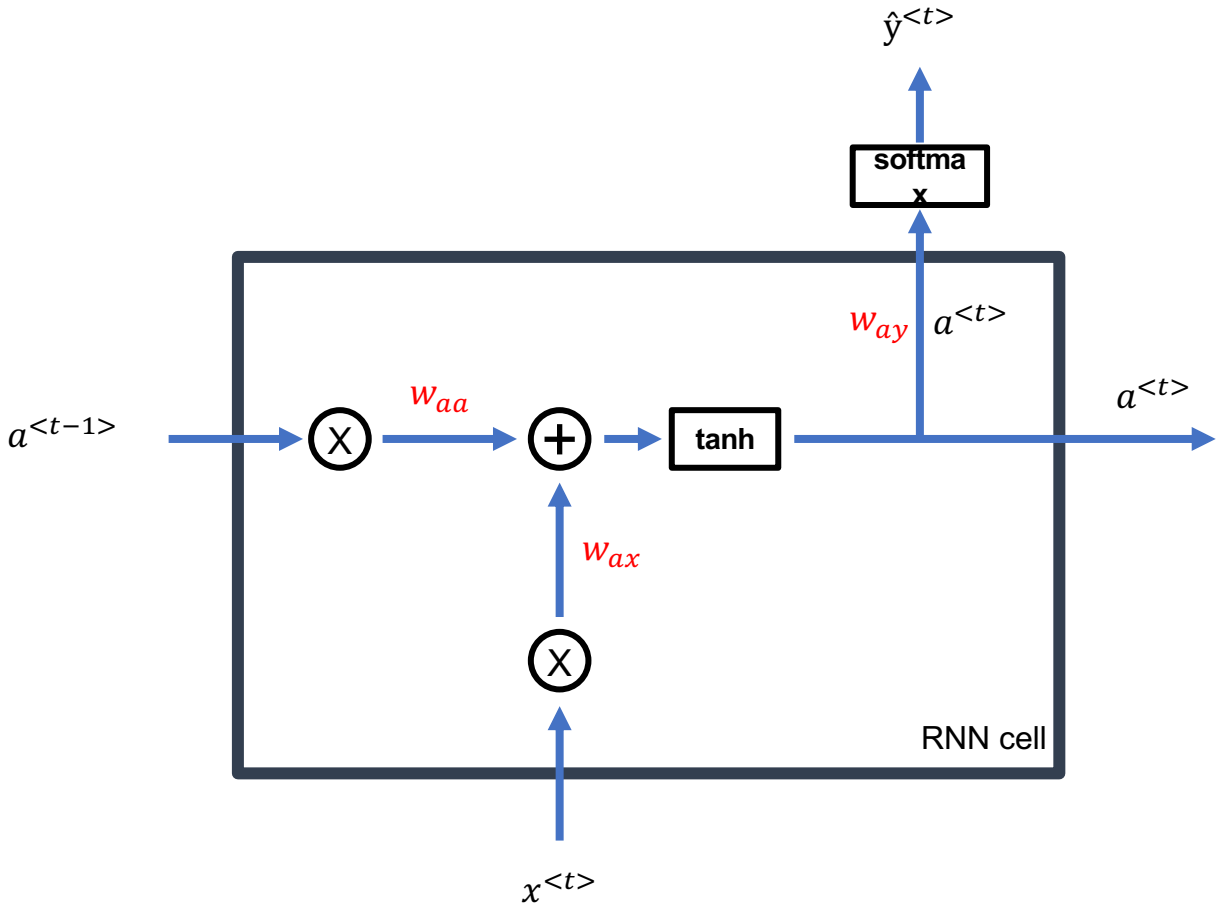
- RNN
- Seq2Seq
- Attention
- Implement


2014 Seq2seq、GRU
1. Learning phrase representations
2. Seq2Seq Learning with NN
2015
3. Neural Machine Translation
Attention 出現
2015
4. Effective Approaches to Attention-based
Attention 概念延伸
2017
5.Convolutional Seq2seq Learning
Cnn 取代 Rnn，做Attention
2017 Self-Attention取代Cnn、Rnn
6. Attention is all you need
7. Atrank: An Attention-Based User Behavior

Close Look at RNN Cell

Recurrent Neural Network



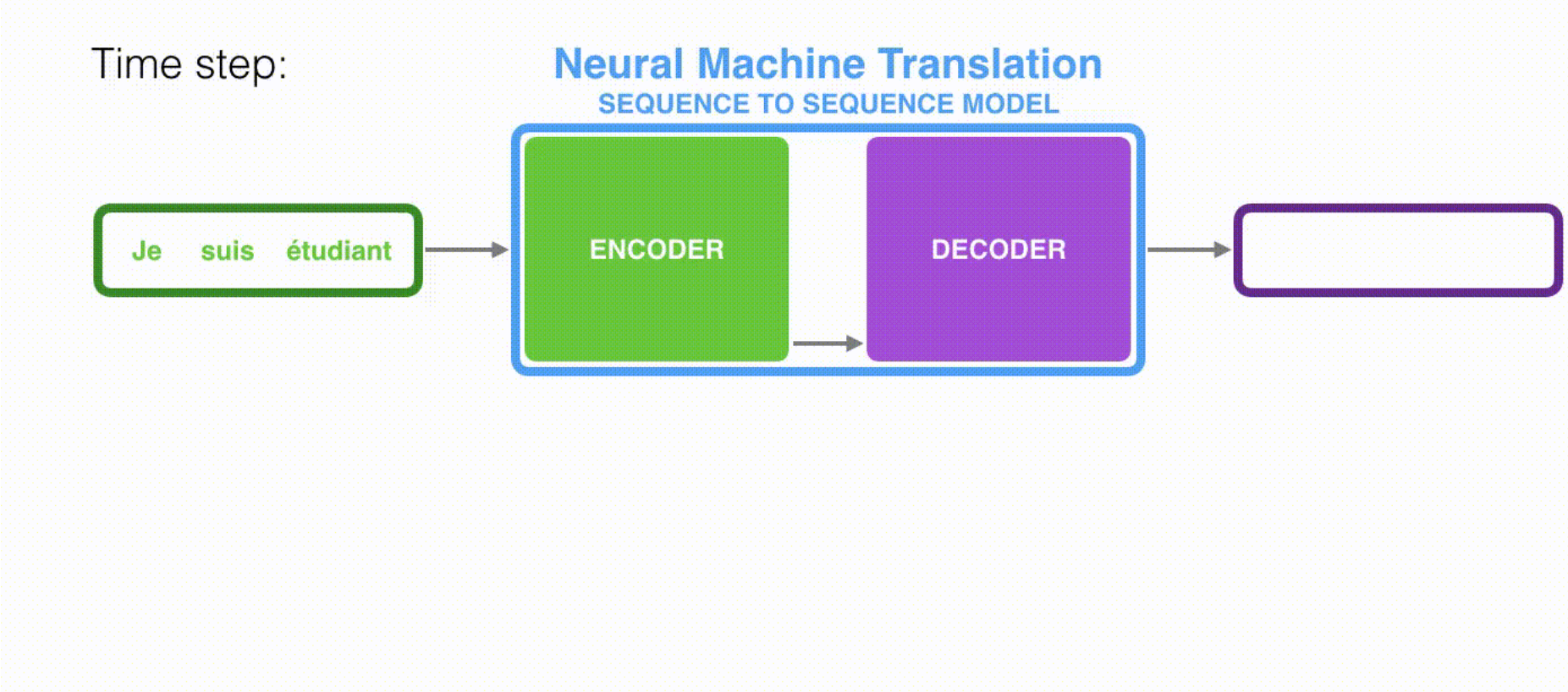


$$a^{<t>} = \tanh(W_{ax}x^{<t>} + W_{aa}a^{<t-1>} + b_a)$$

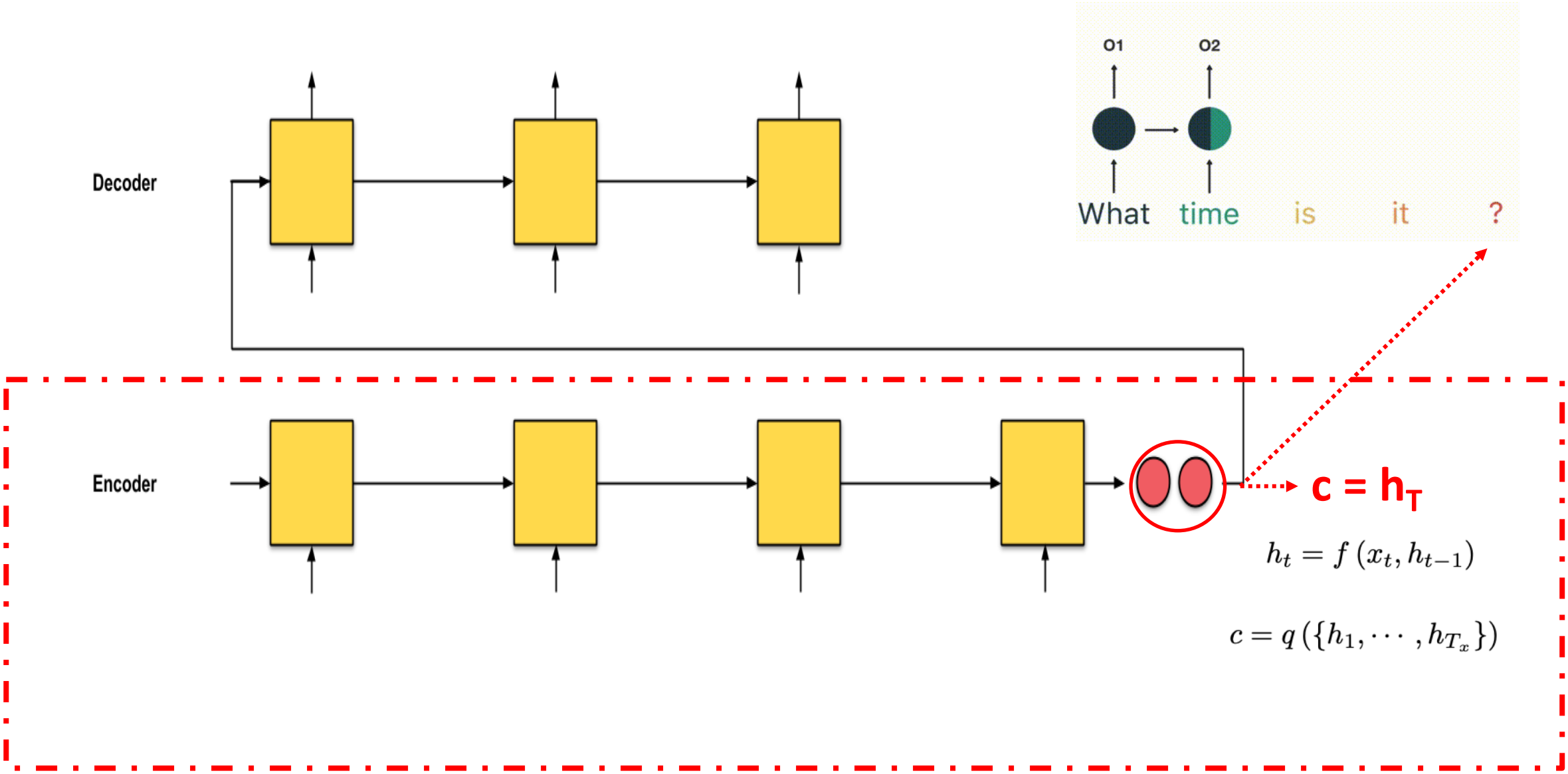
$$\hat{y}^{<t>} = \text{softmax}(W_{ay}a^{<t>} + b_y)$$

Seq2Seq

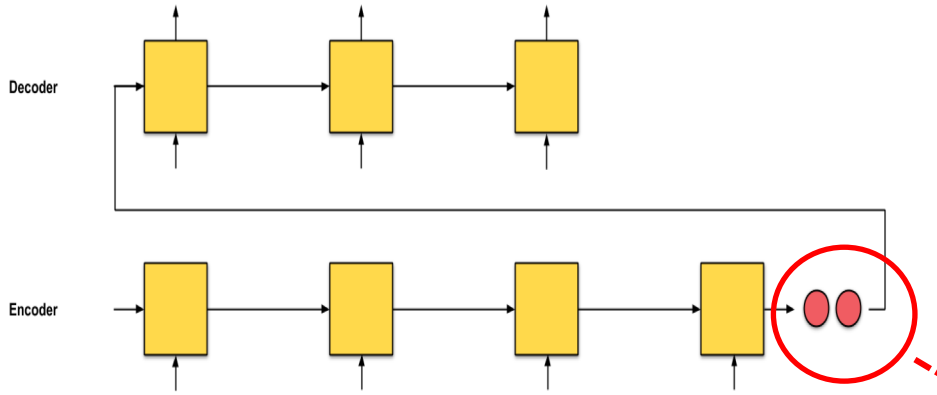
RNN input:長度不固定 ouput:固定

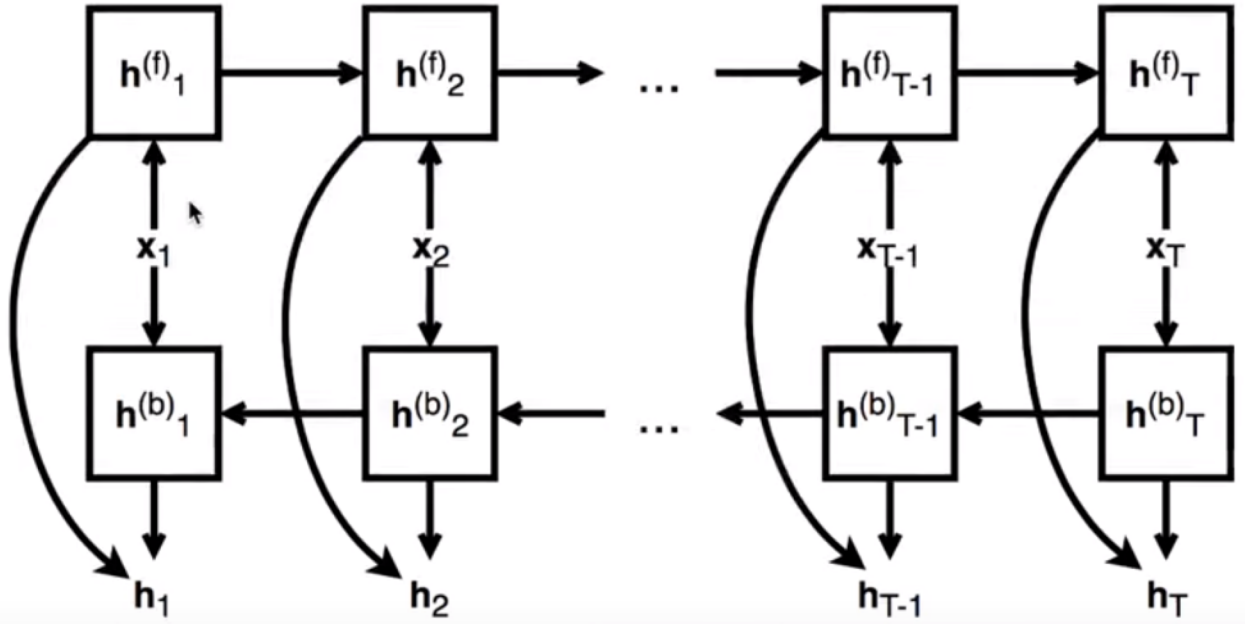


Seq2Seq

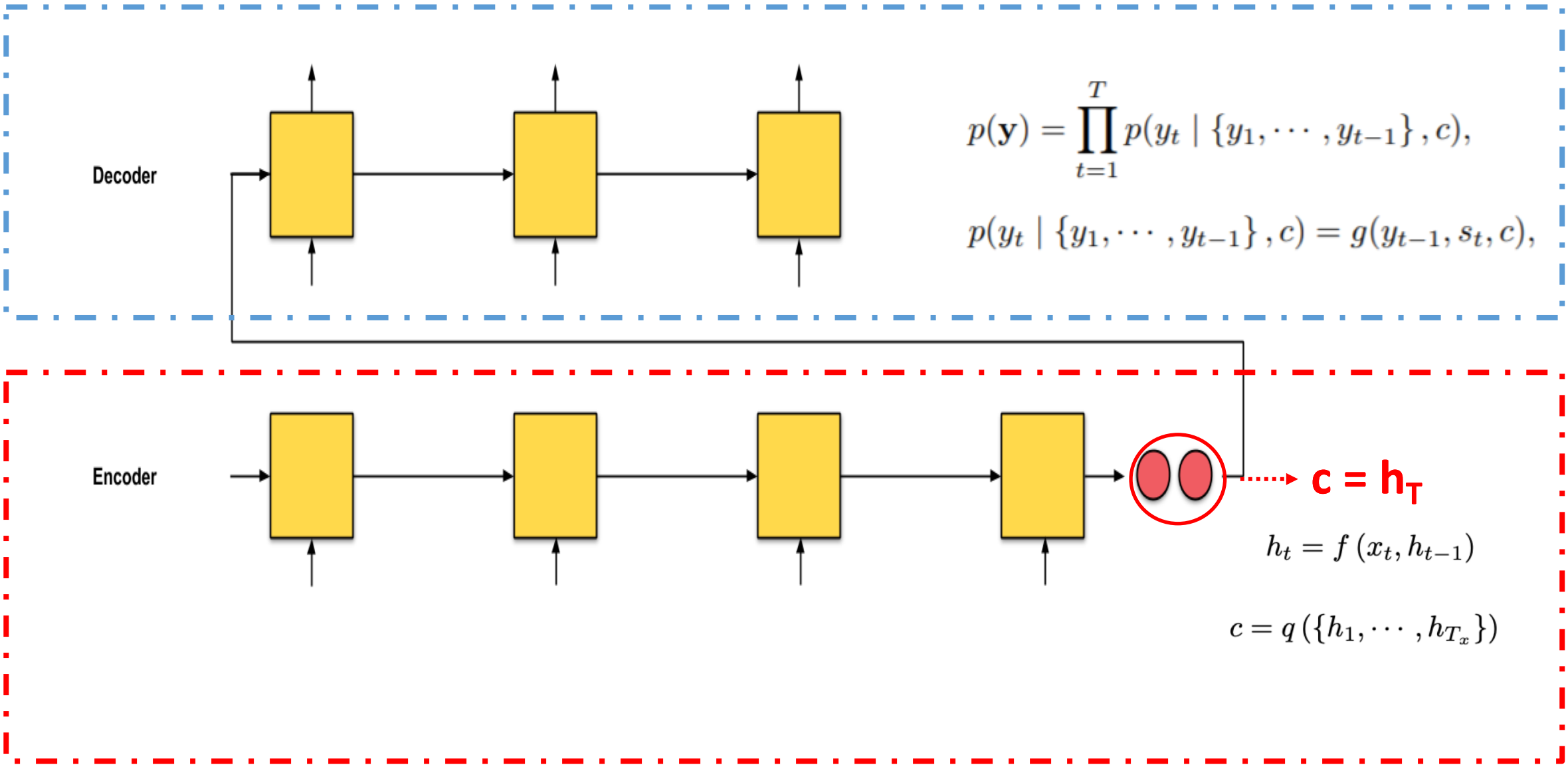

Decoder
Encoder
o1
o2
What
time
is
it
?
$c = h_T$
$h_t = f(x_t, h_{t-1})$
$c = q(\{h_1, \cdots, h_{T_x}\})$

Seq2Seq


Decoder
Encoder


h(f)1
h(f)2
...
h(f)T-1
h(f)T
x1
x2
xT-1
xT
h(b)1
h(b)2
...
h(b)T-1
h(b)T
h1
h2
hT-1
hT

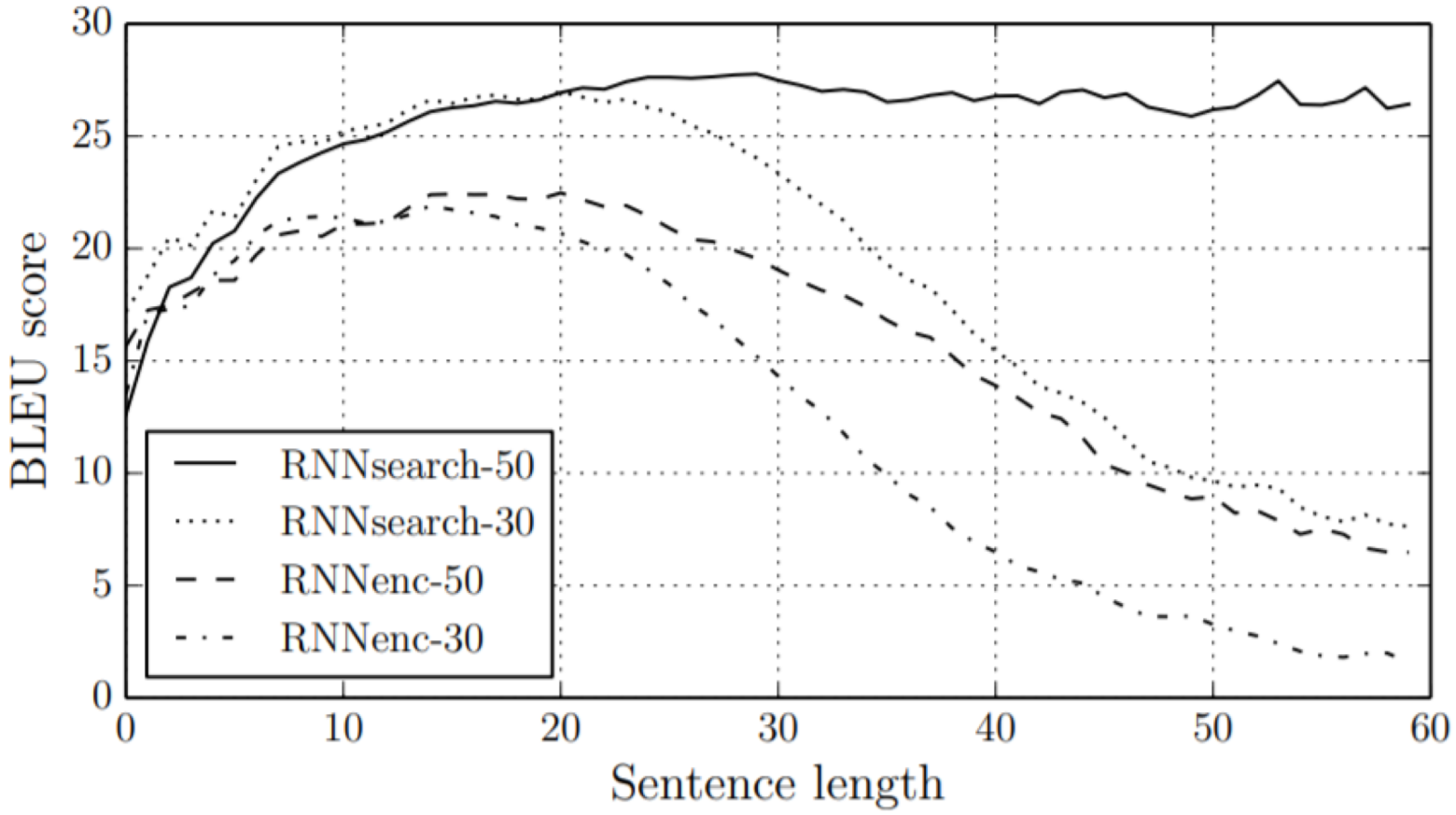
Seq2Seq


Decoder
Encoder
$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c),$
$p(y_t \mid \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c),$
$c = h_T$
$h_t = f(x_t, h_{t-1})$
$c = q(\{h_1, \cdots, h_{T_x}\})$

Seq2Seq bottleneck

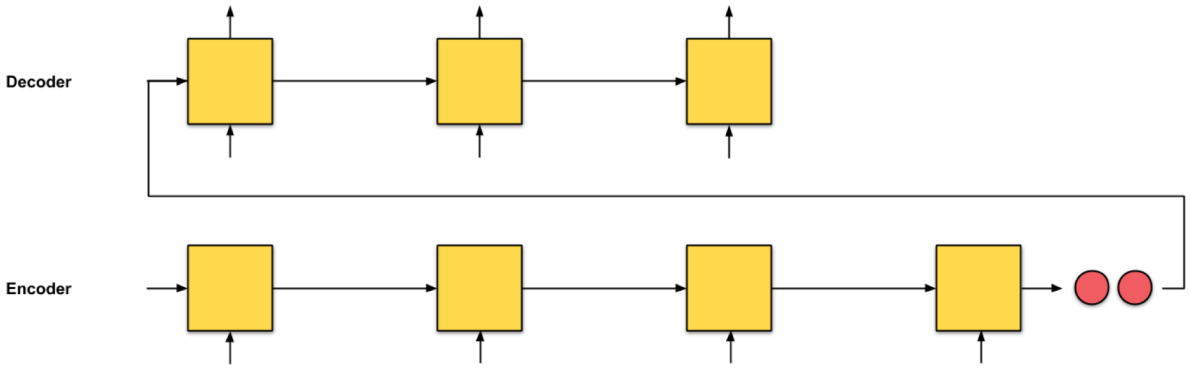
1、把輸入X 所有訊息壓縮到一個固定長度的hidden state C。當輸入句子長度很長，模型的效能急劇下降。

2、把輸入X encoder成一個固定的長度，對於句子中每個詞都賦予相同的權重，這樣做是不合理。

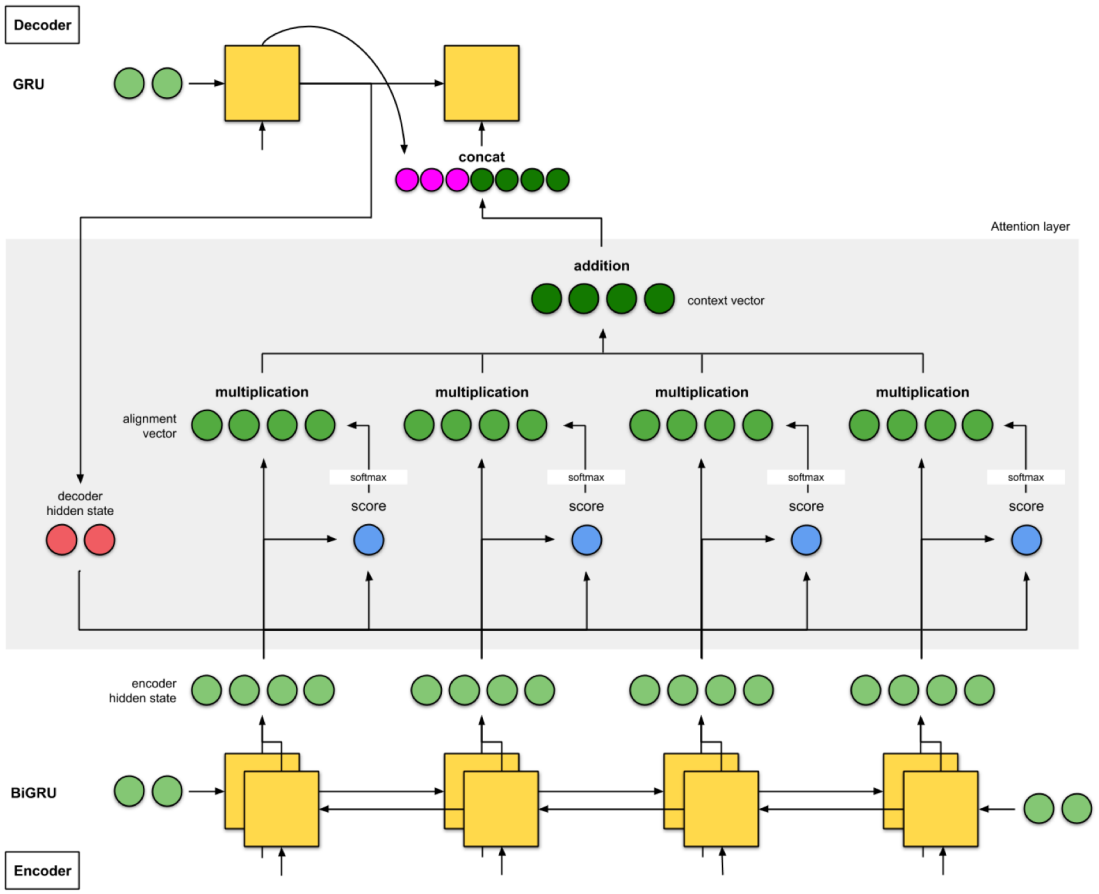


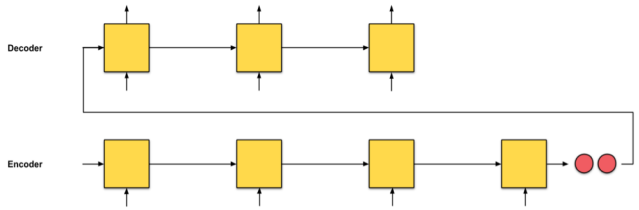
Attention

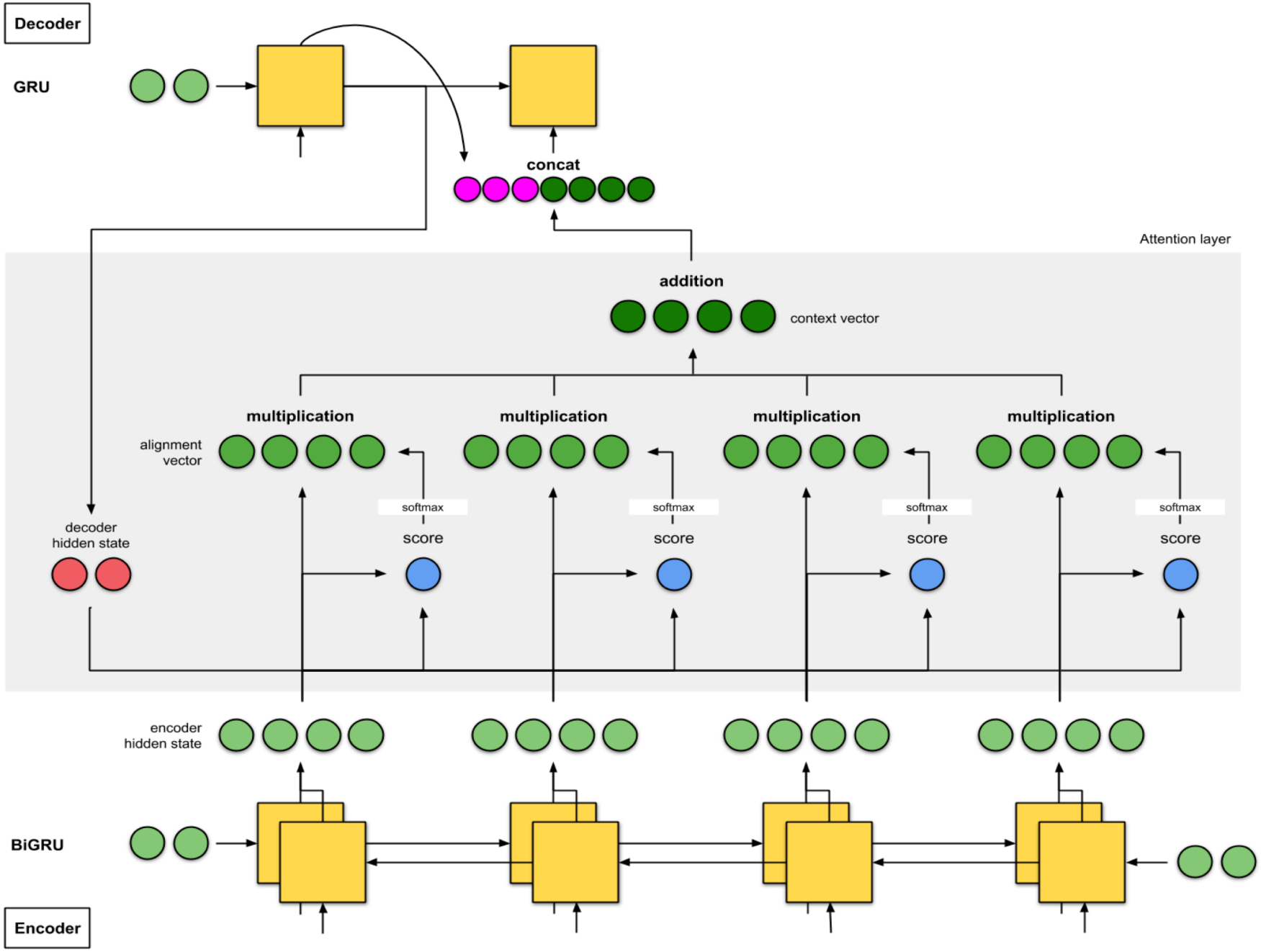
Seq2seq



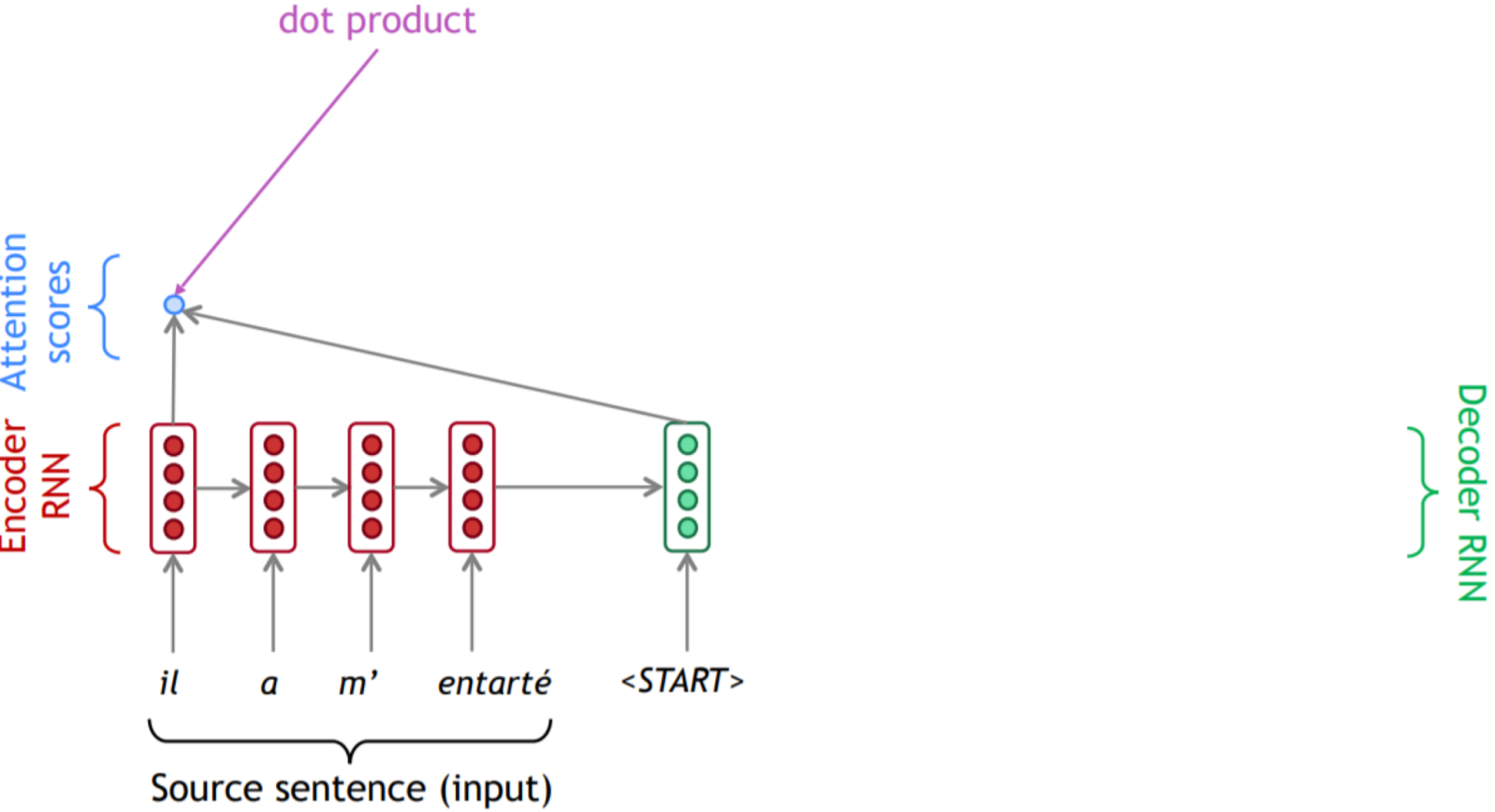
Seq2seq+Attention



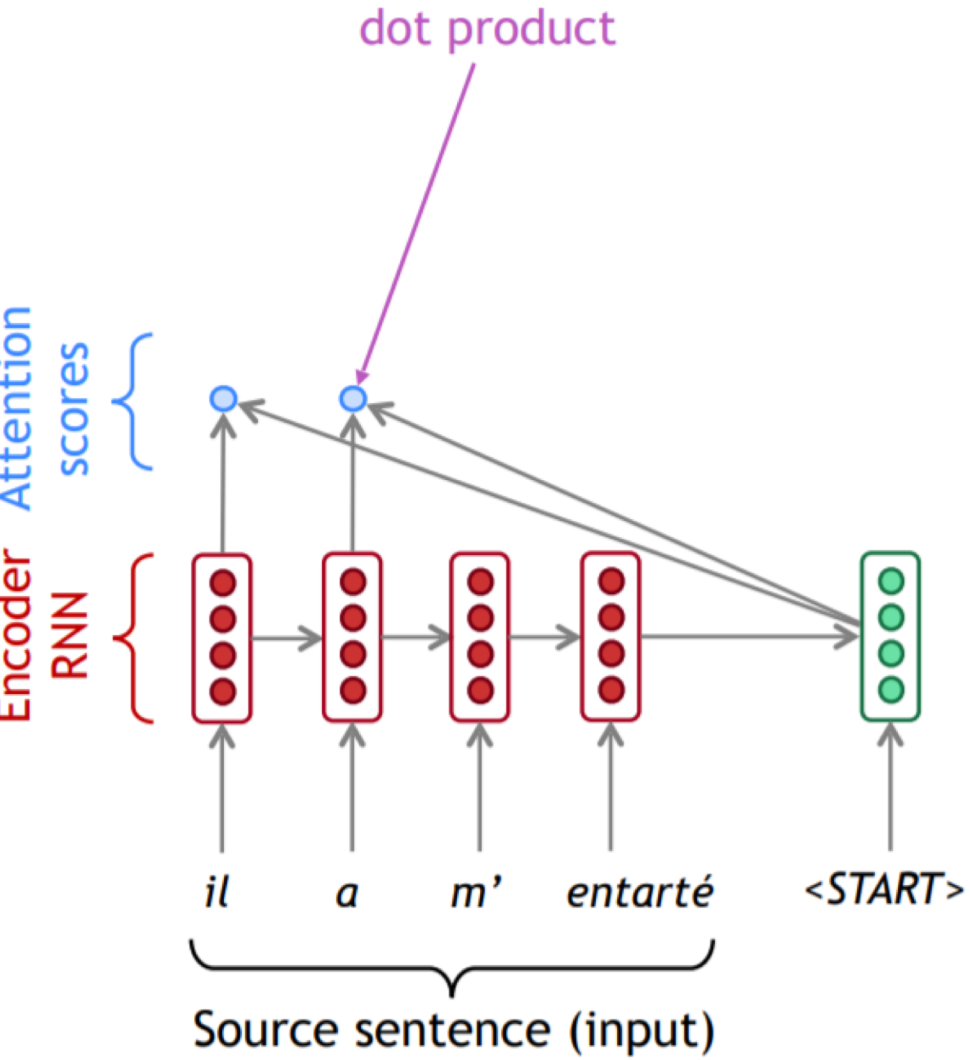

Decoder
Encoder


Decoder
GRU
concat
Attention layer
addition
context vector
multiplication
multiplication
multiplication
multiplication
alignment vector
softmax
softmax
softmax
softmax
score
score
score
score
decoder hidden state
encoder hidden state
BiGRU
Encoder

Attention

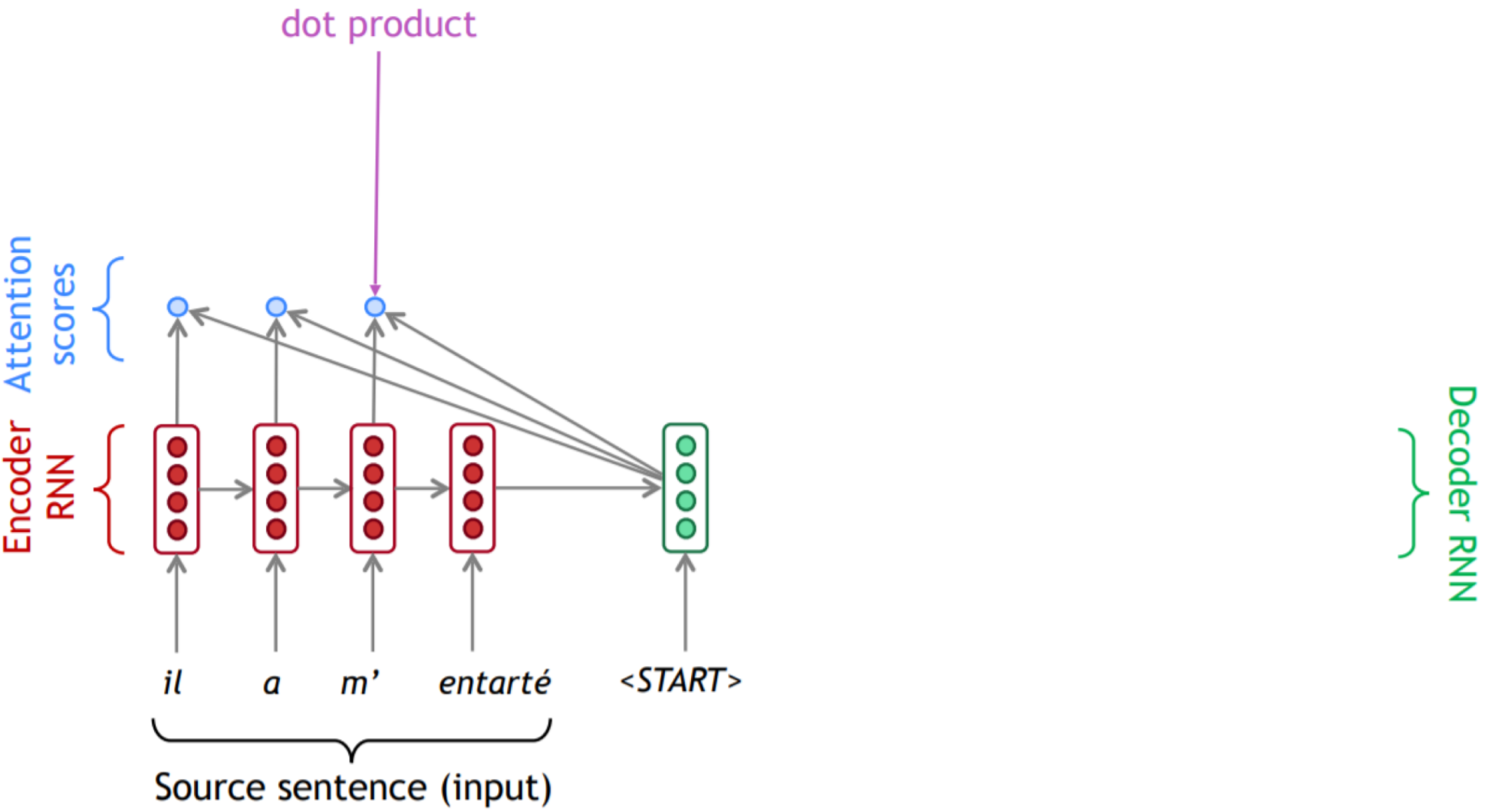

dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)

Attention

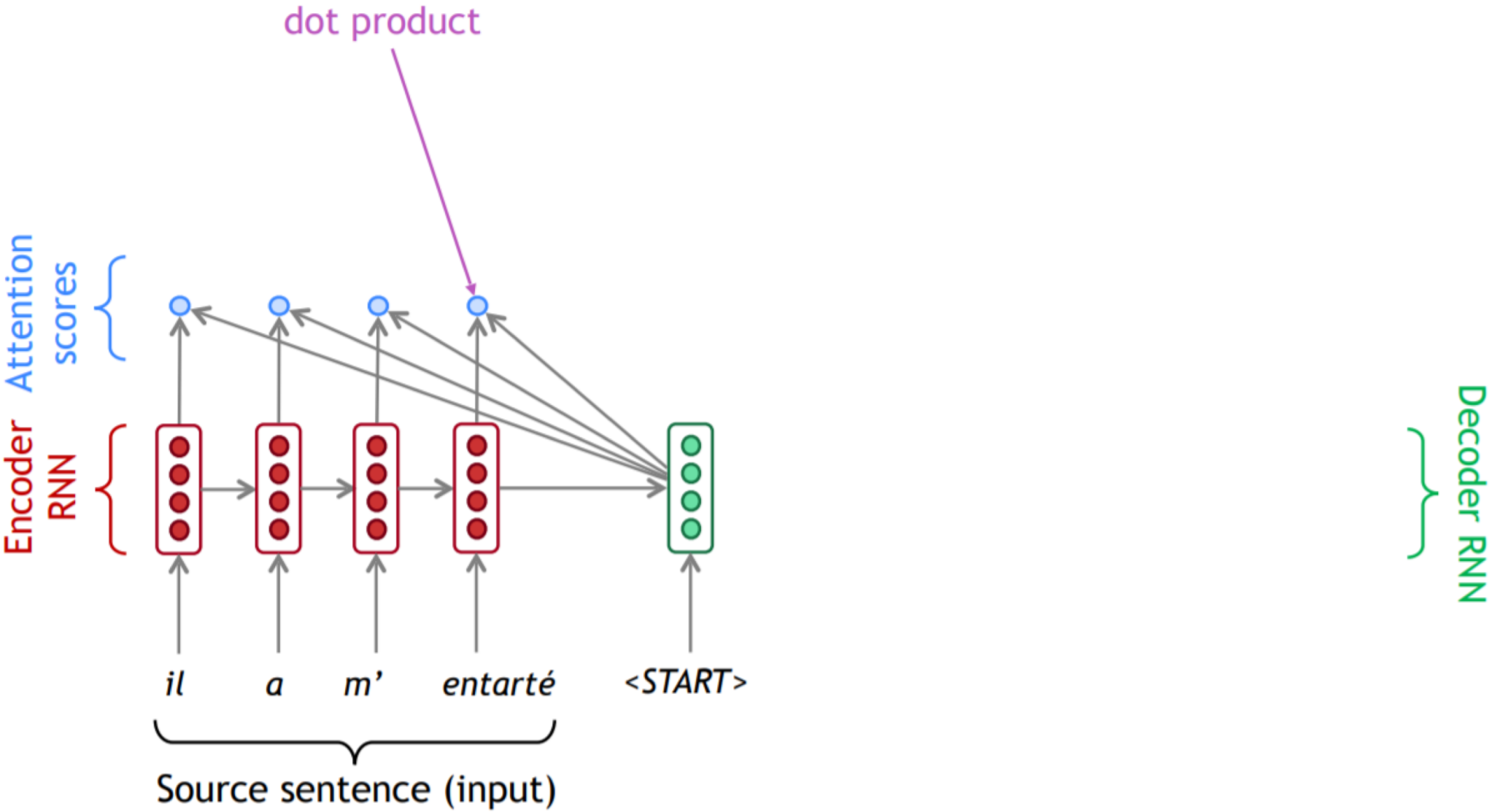


Decoder RNN

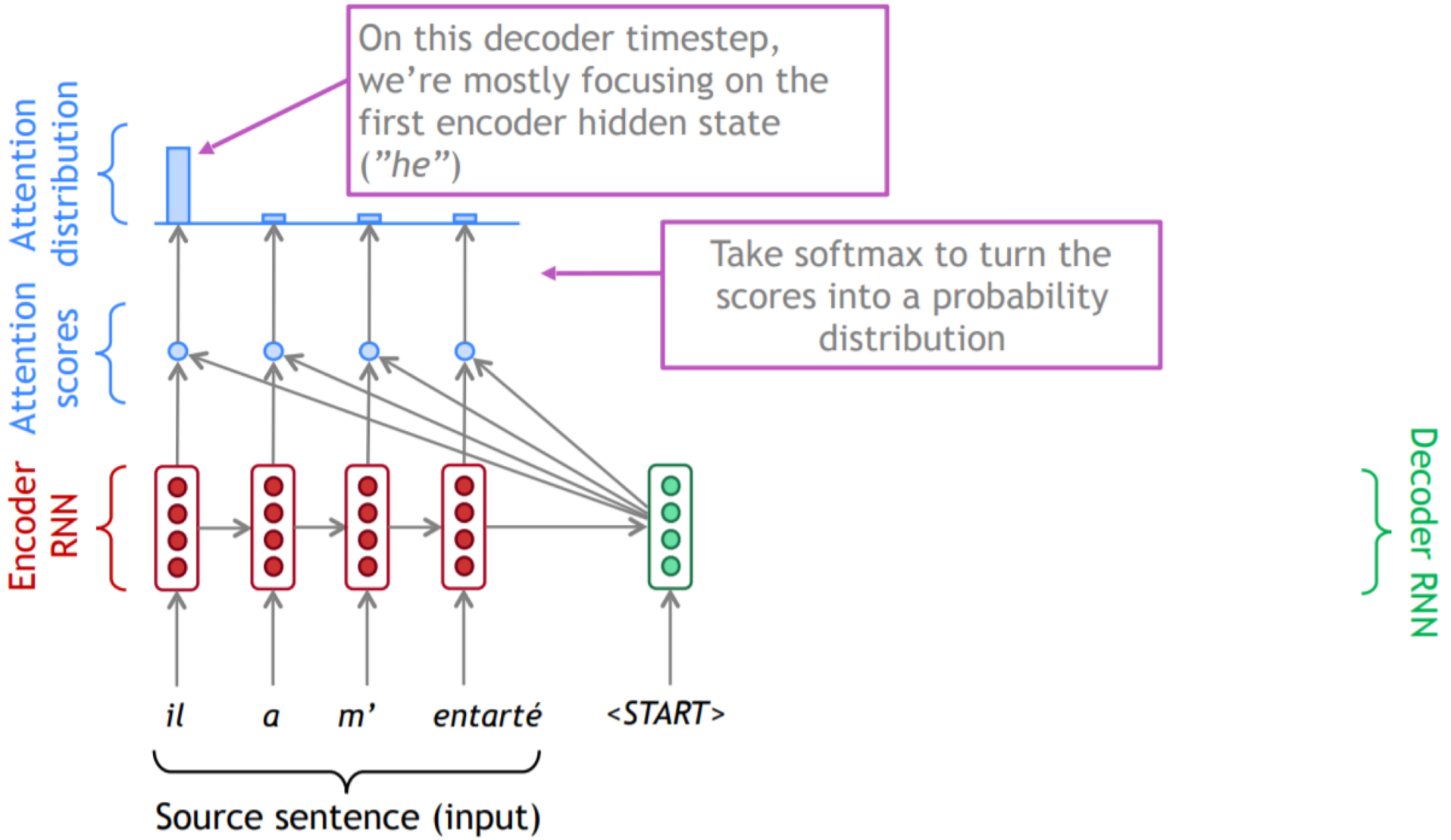
Attention


dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)

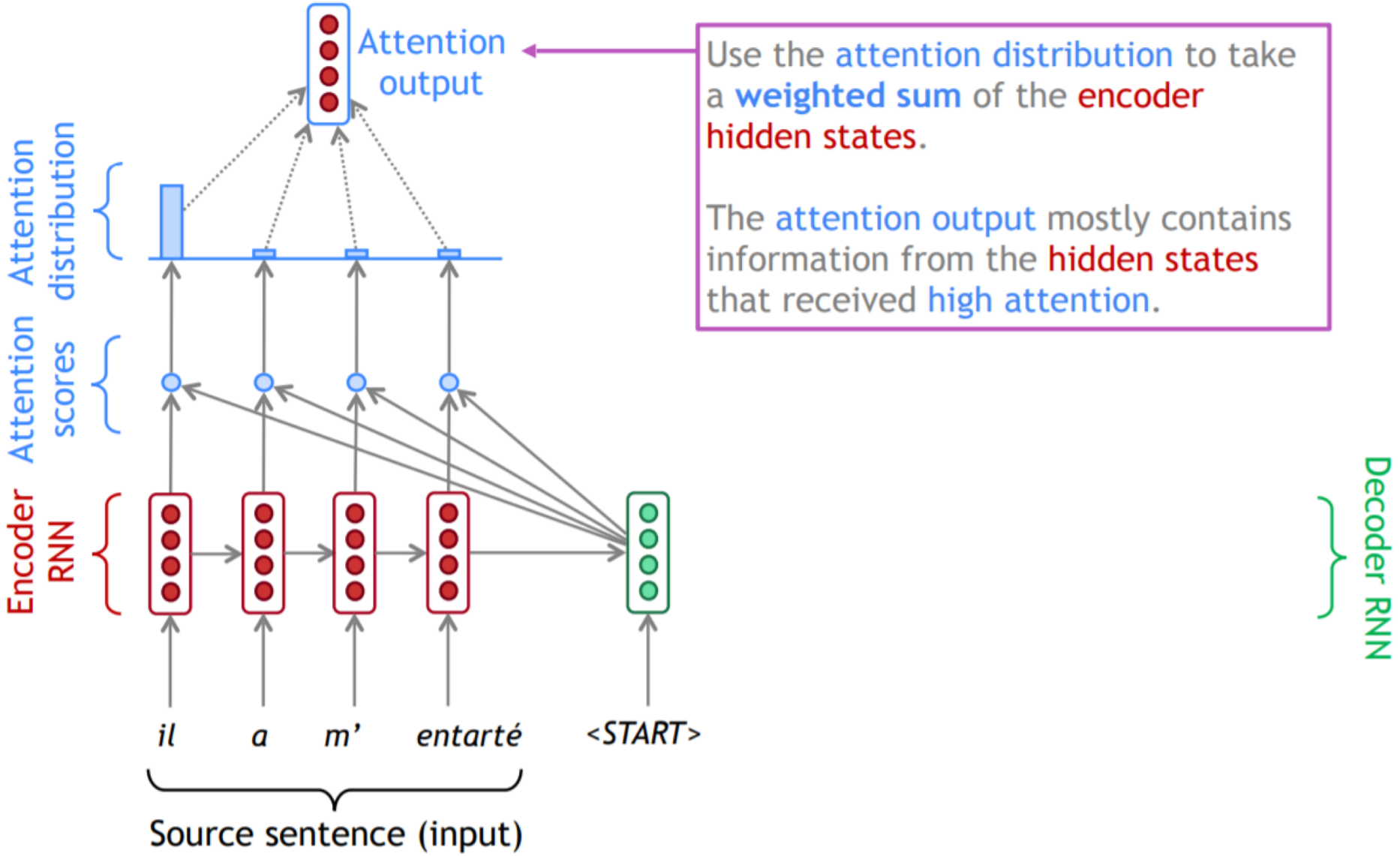
Attention


dot product
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)

Attention


On this decoder timestep, we're mostly focusing on the first encoder hidden state ("he")
Take softmax to turn the scores into a probability distribution
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)

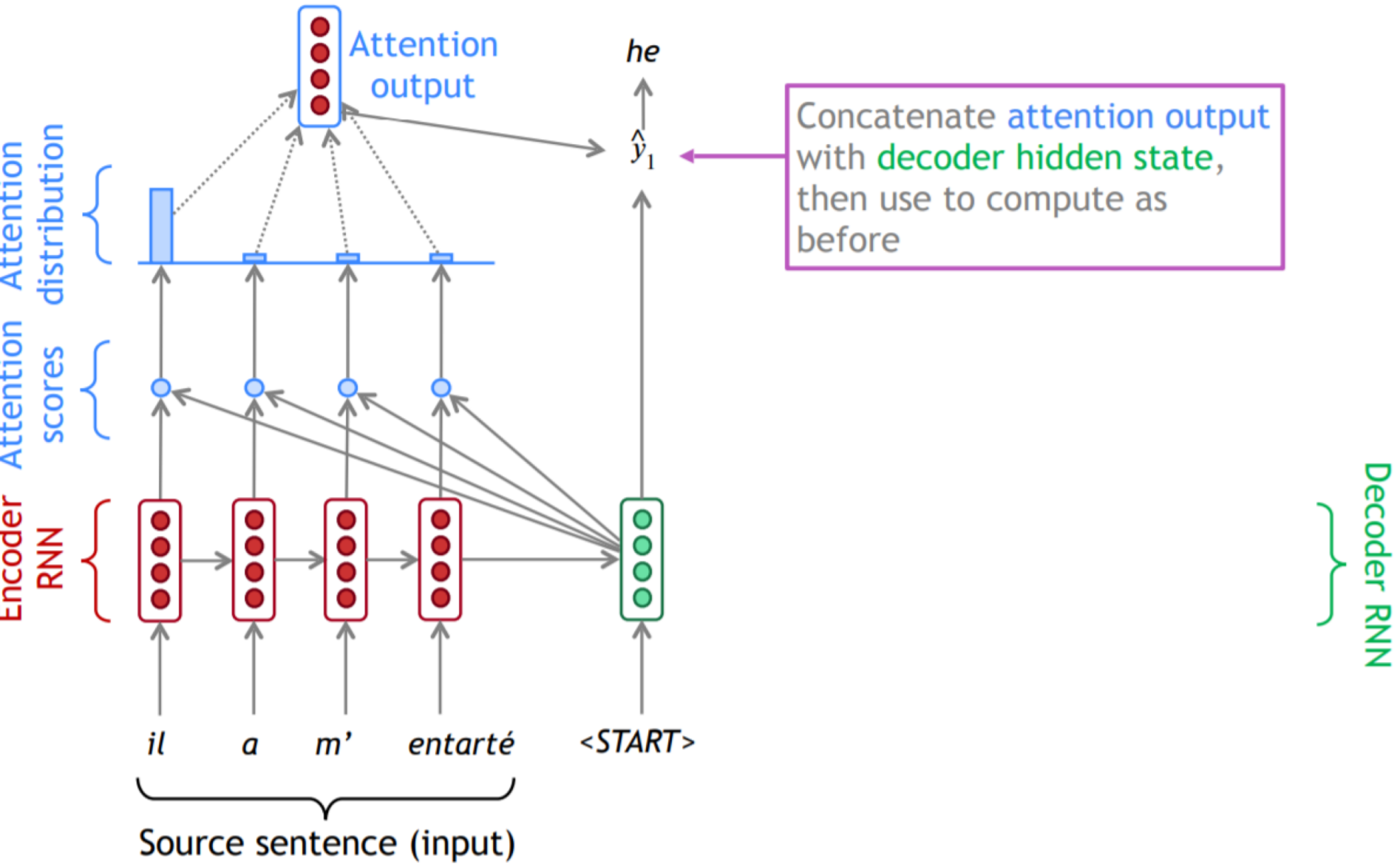
Attention



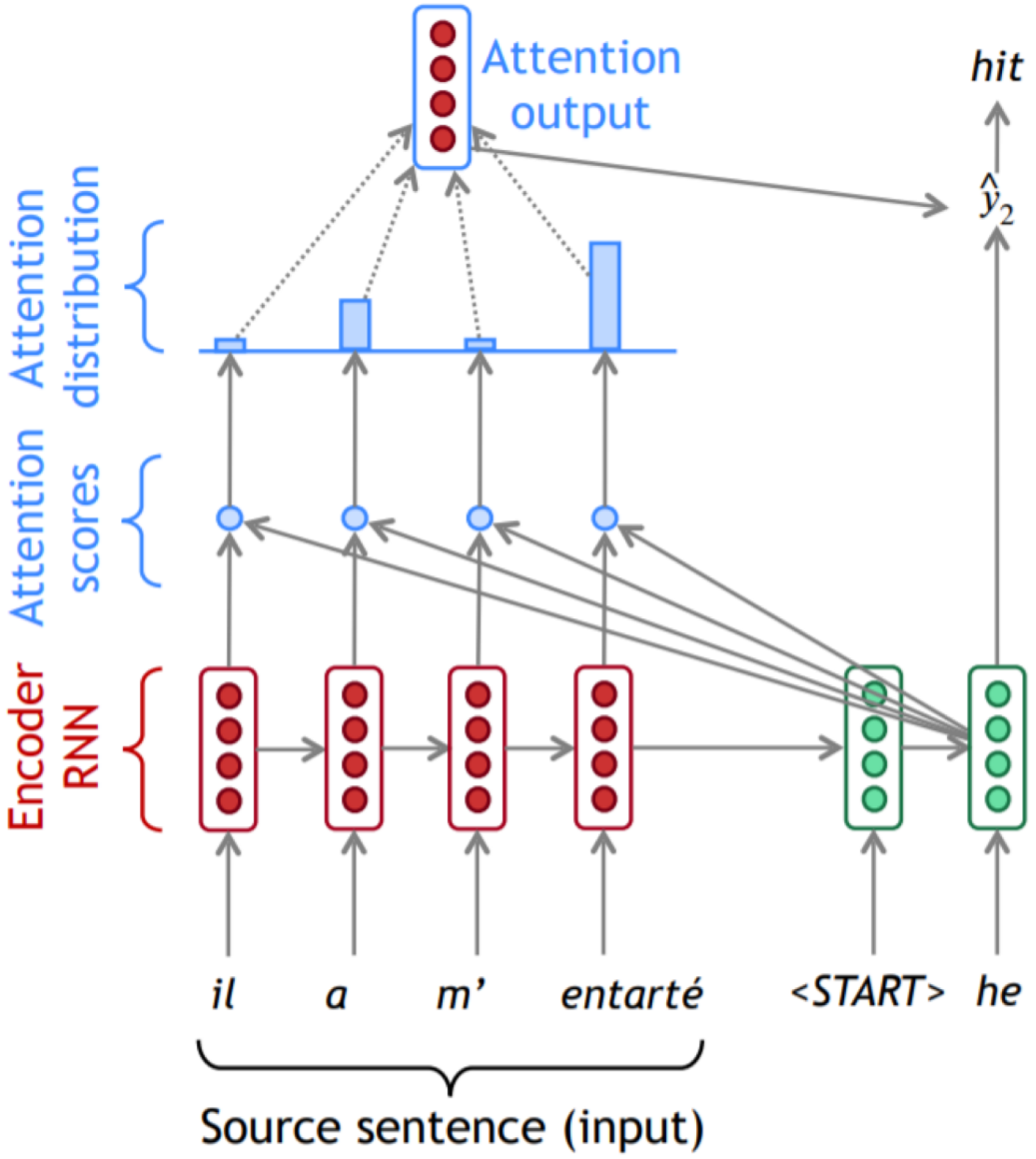
Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information from the hidden states that received high attention.

Attention


Attention output
he
$\hat{y}_1$
Concatenate attention output with decoder hidden state, then use to compute as before
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
Source sentence (input)

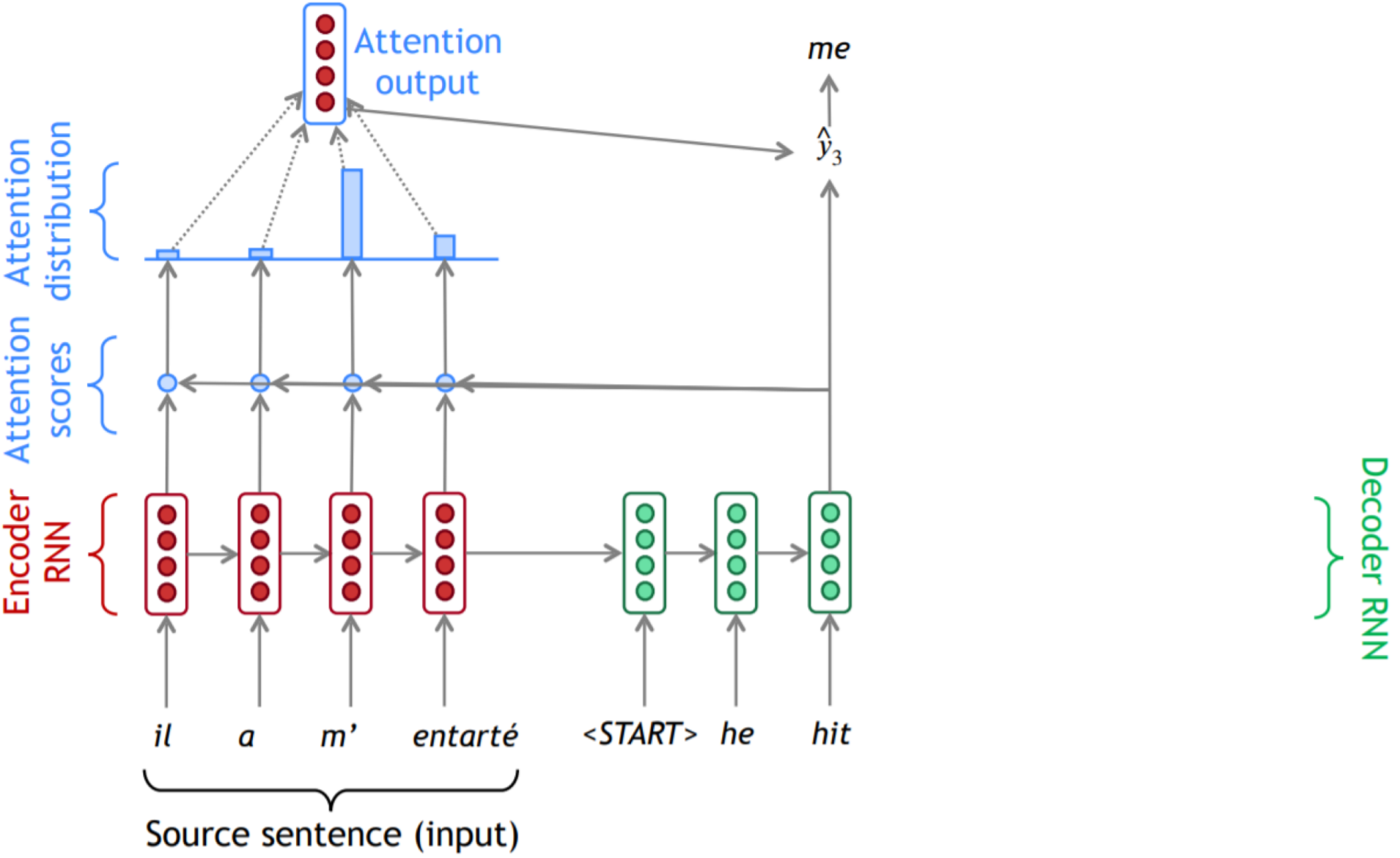
Attention



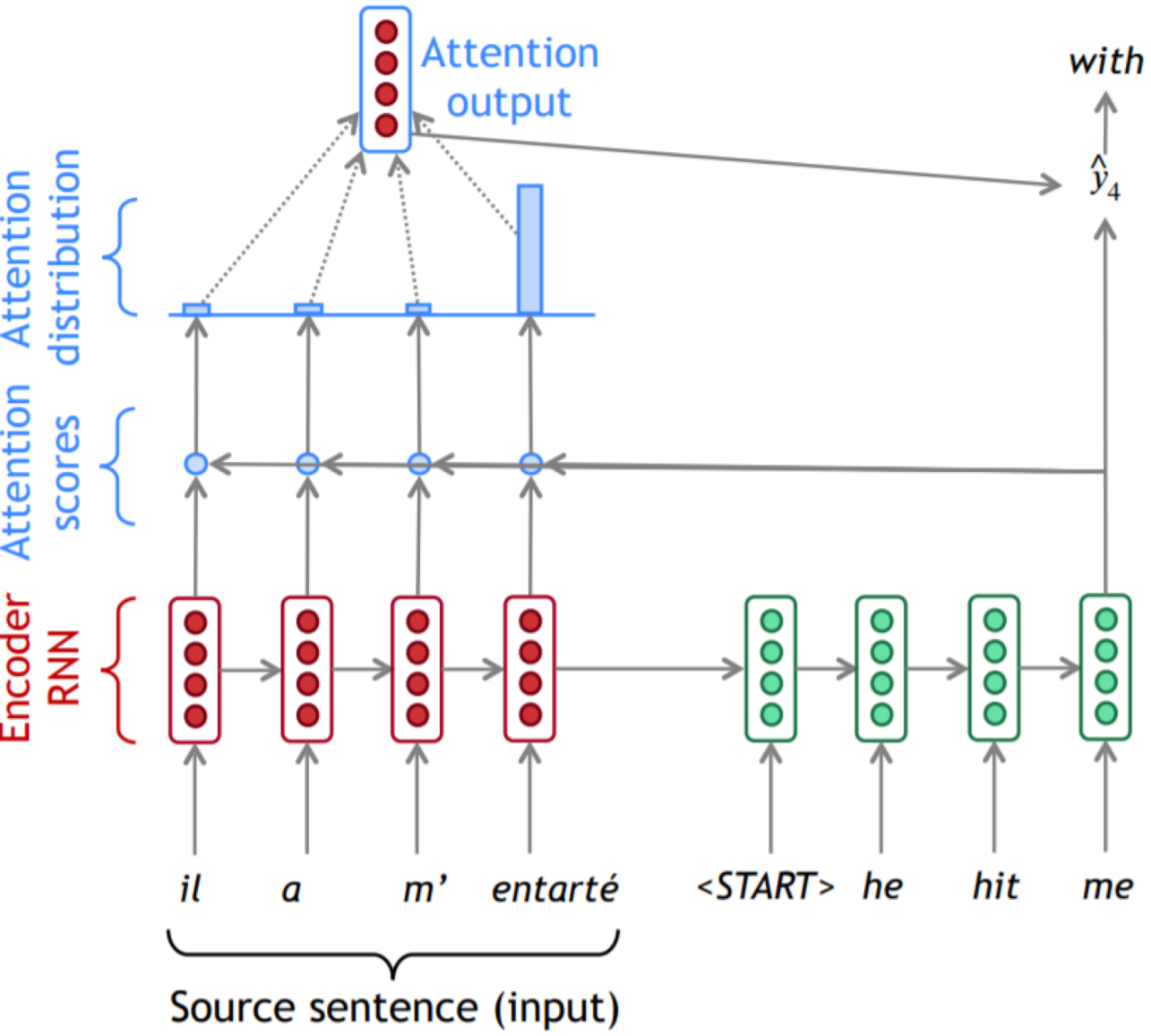
Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input).

Decoder RNN

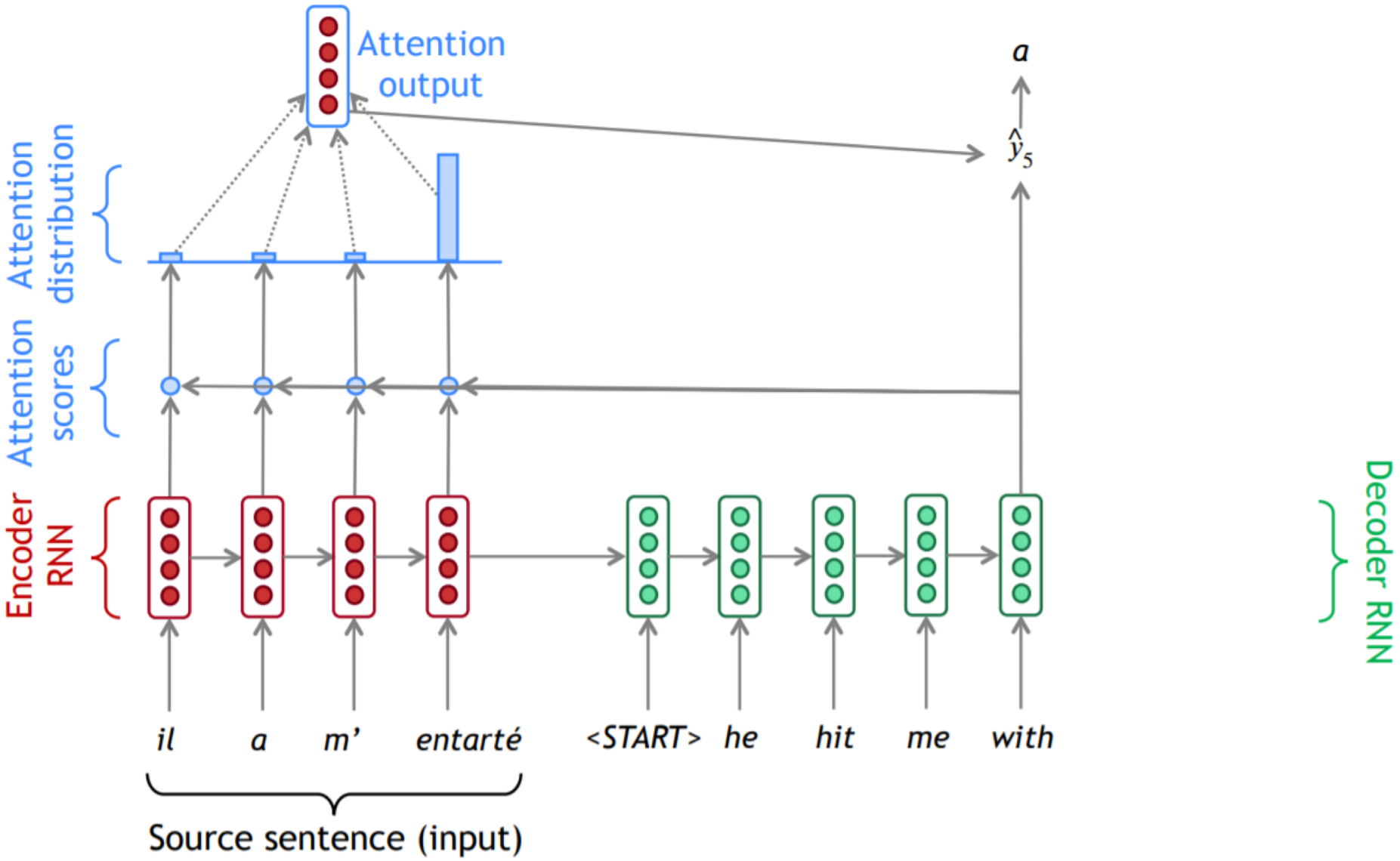
Attention


Attention output
me
$\hat{y}_3$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
Source sentence (input)

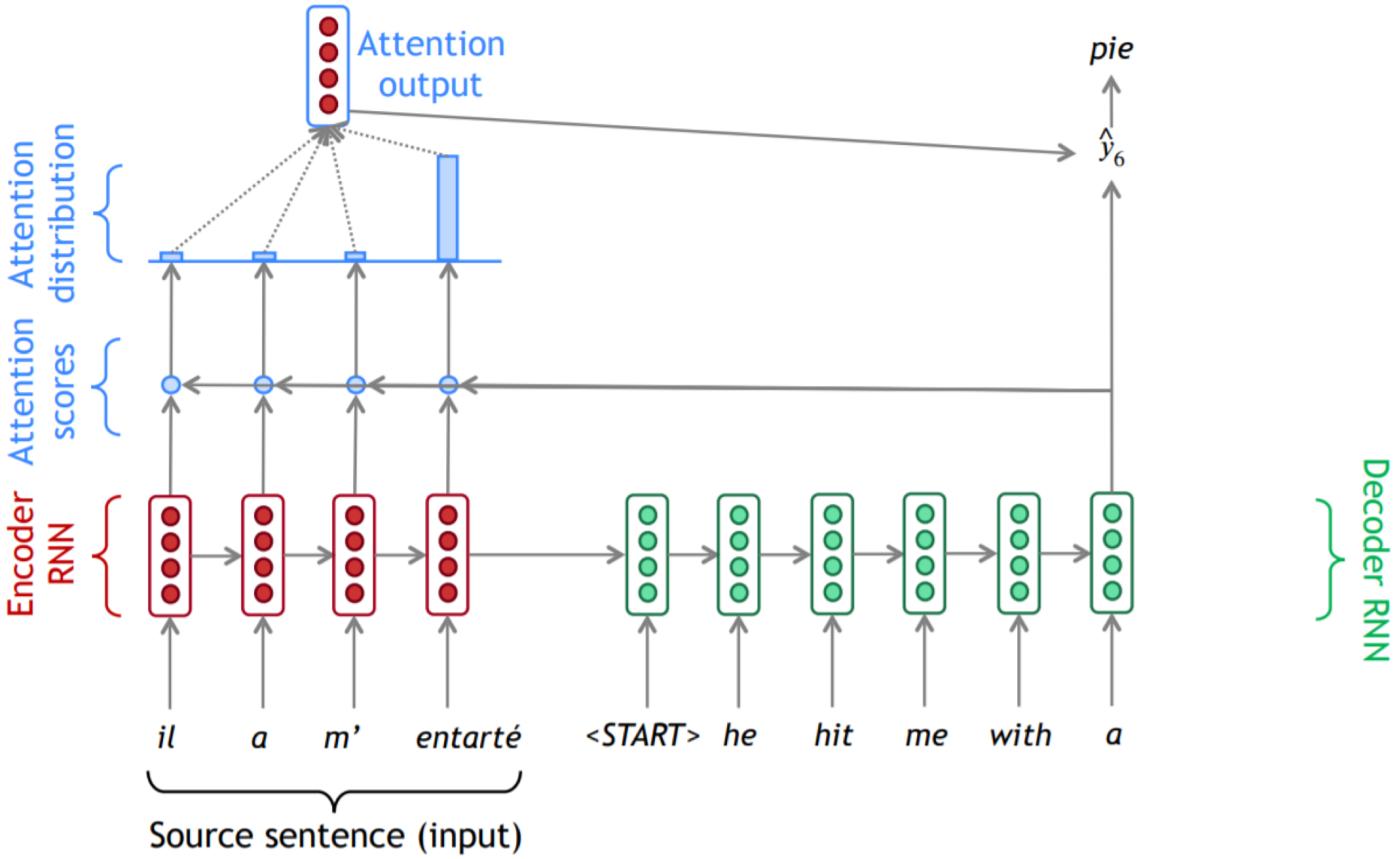
Attention


Attention output
with
$\hat{y}_4$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
Source sentence (input)

Attention


Attention output
a
$\hat{y}_5$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
Source sentence (input)

Attention


Attention output
pie
$\hat{y}_6$
Attention distribution
Attention scores
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
a
Source sentence (input)

Attention

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$
- On timestep *t*, we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention scores e^t for this step:

$$\boldsymbol{e}^t = [\boldsymbol{s}_t^T \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^T \boldsymbol{h}_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

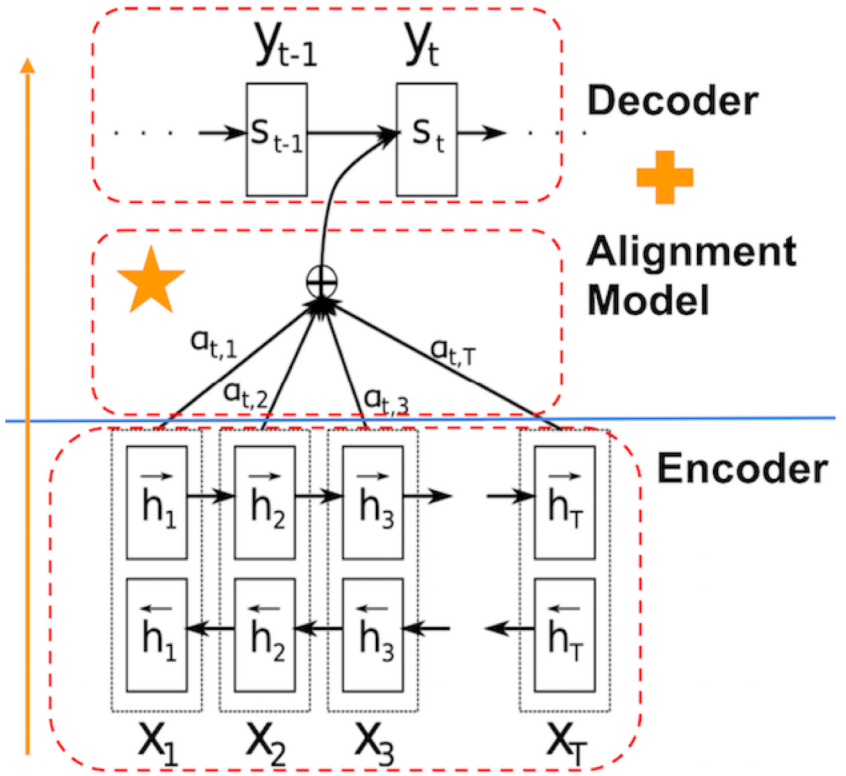
$$\alpha^t = \text{softmax}(\boldsymbol{e}^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output

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

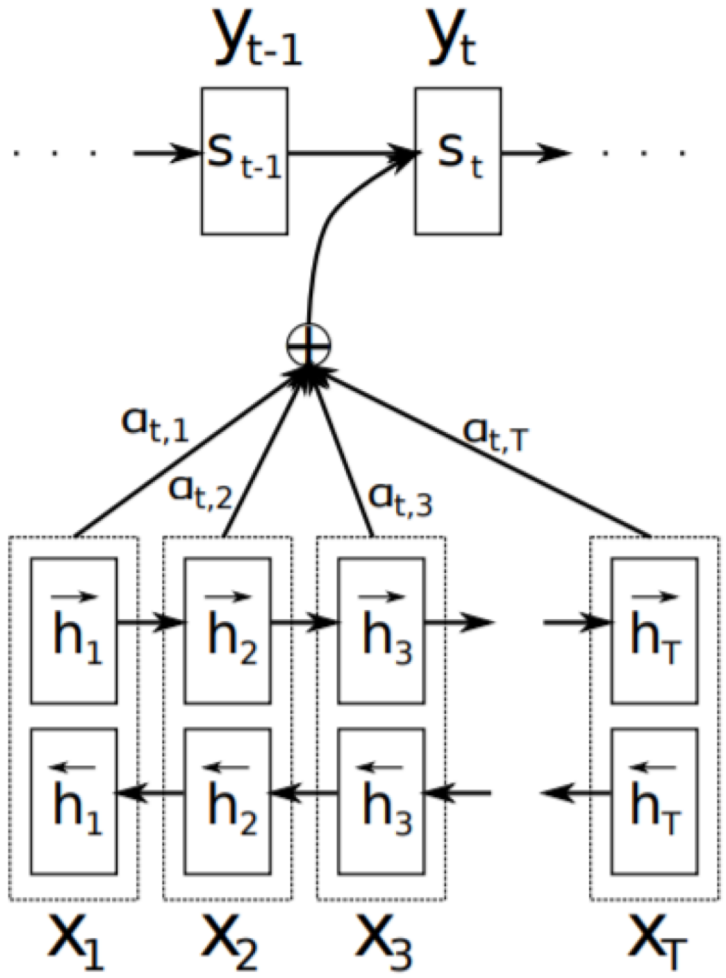
- Finally we concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

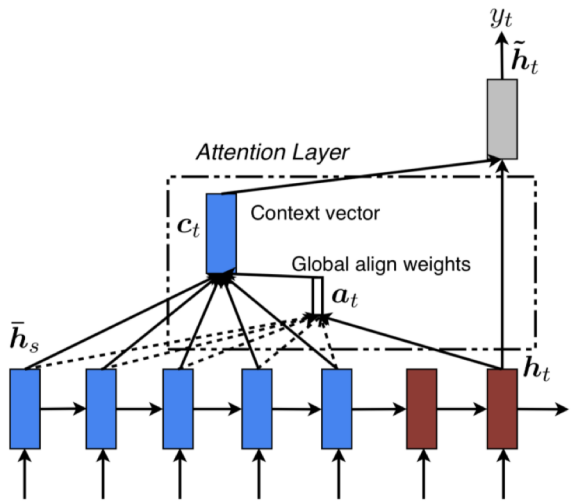


Attention

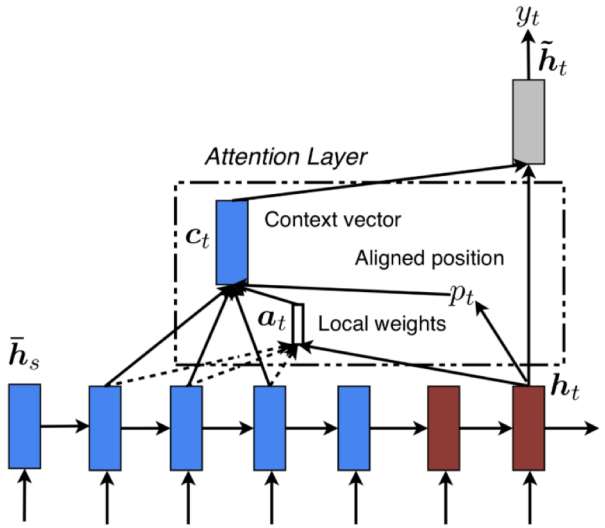
Bahdanau Attention



Luong Attention



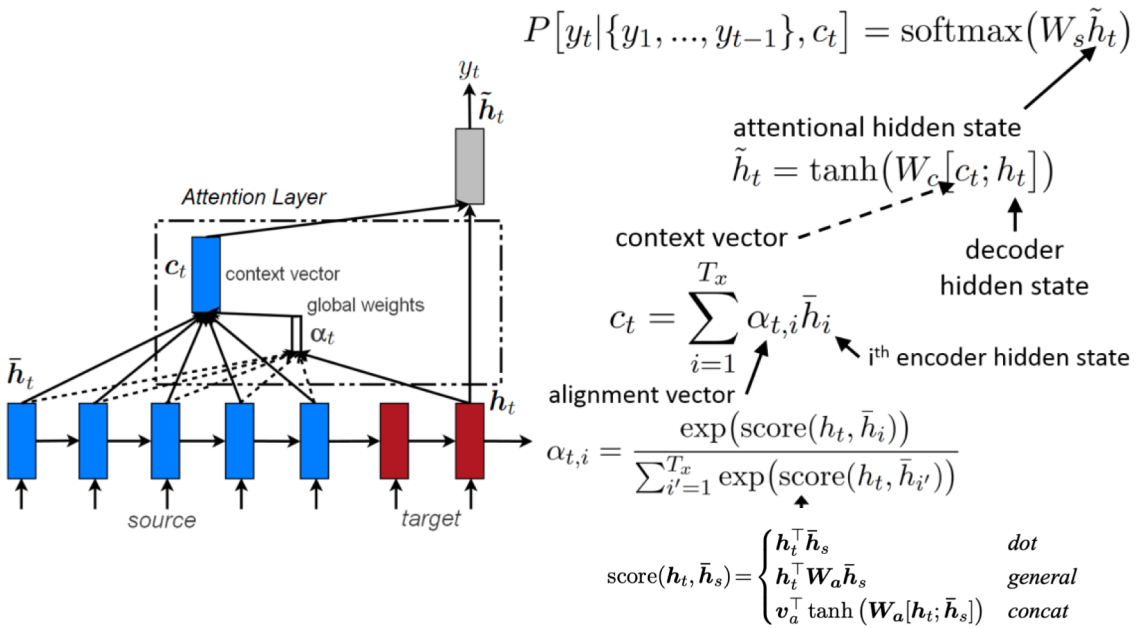
Global Attention Model



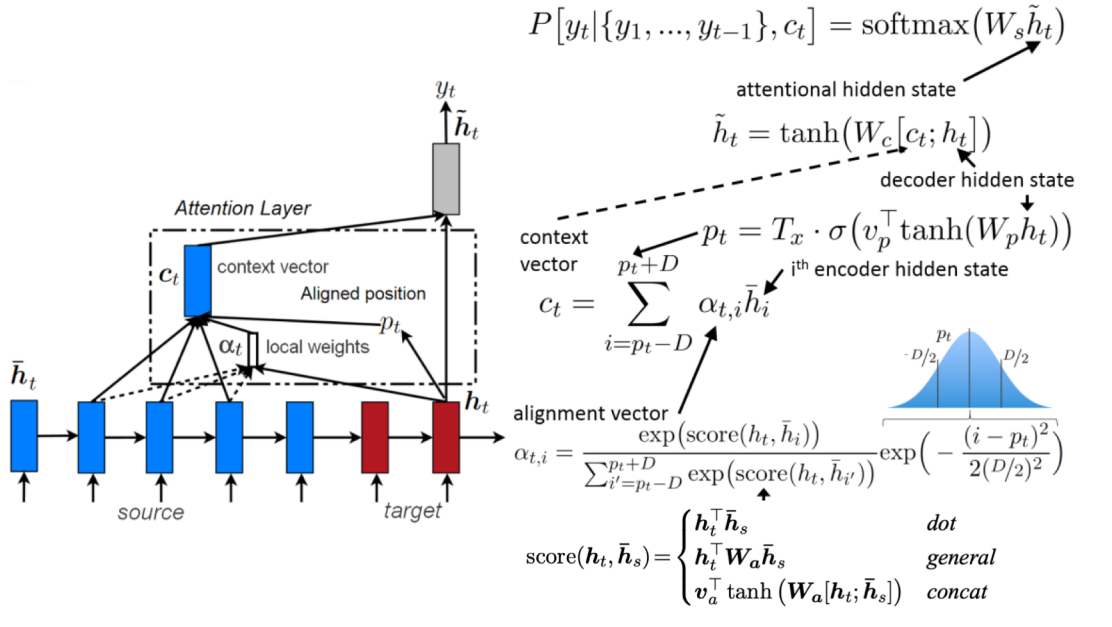
Local Attention Model

Luong Attention

Global Attention



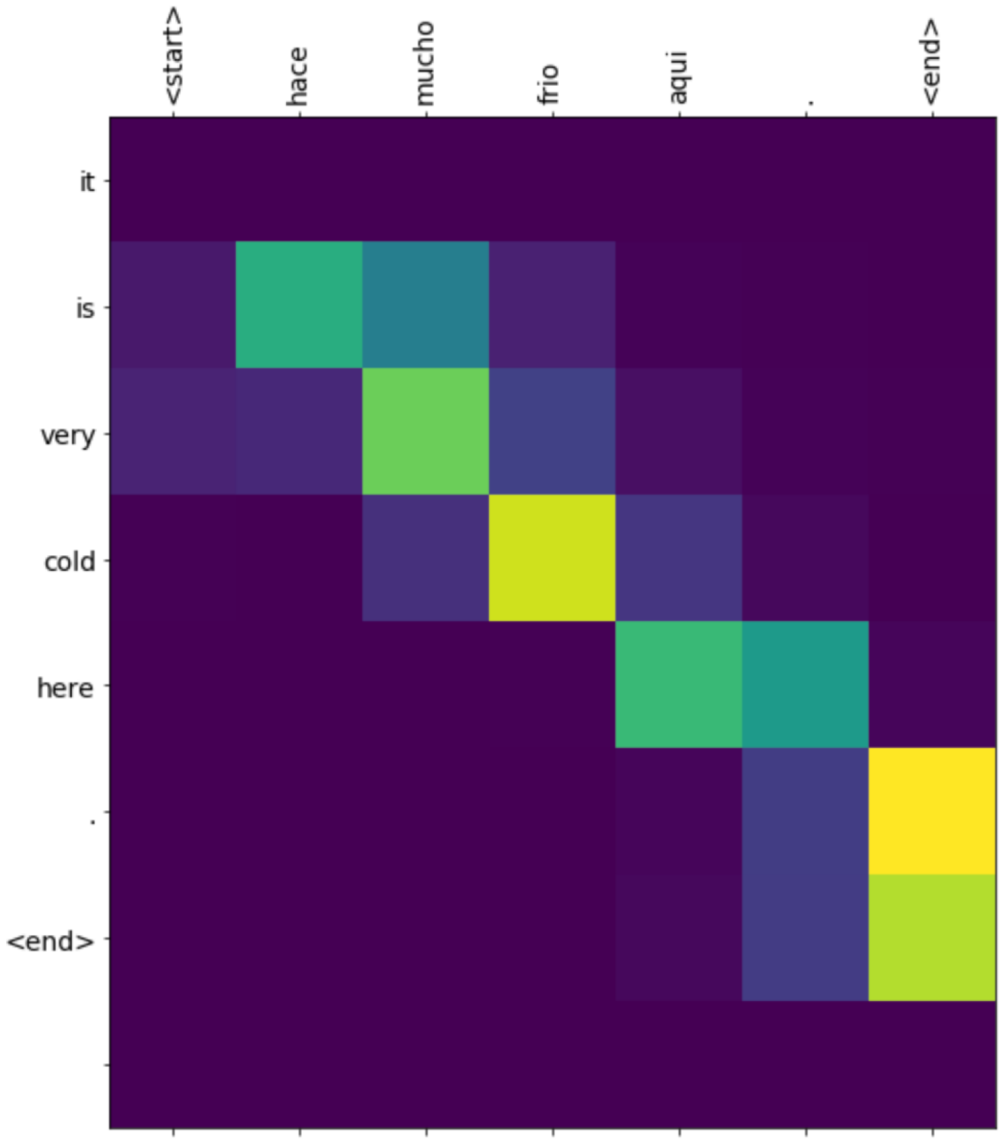
local Attention

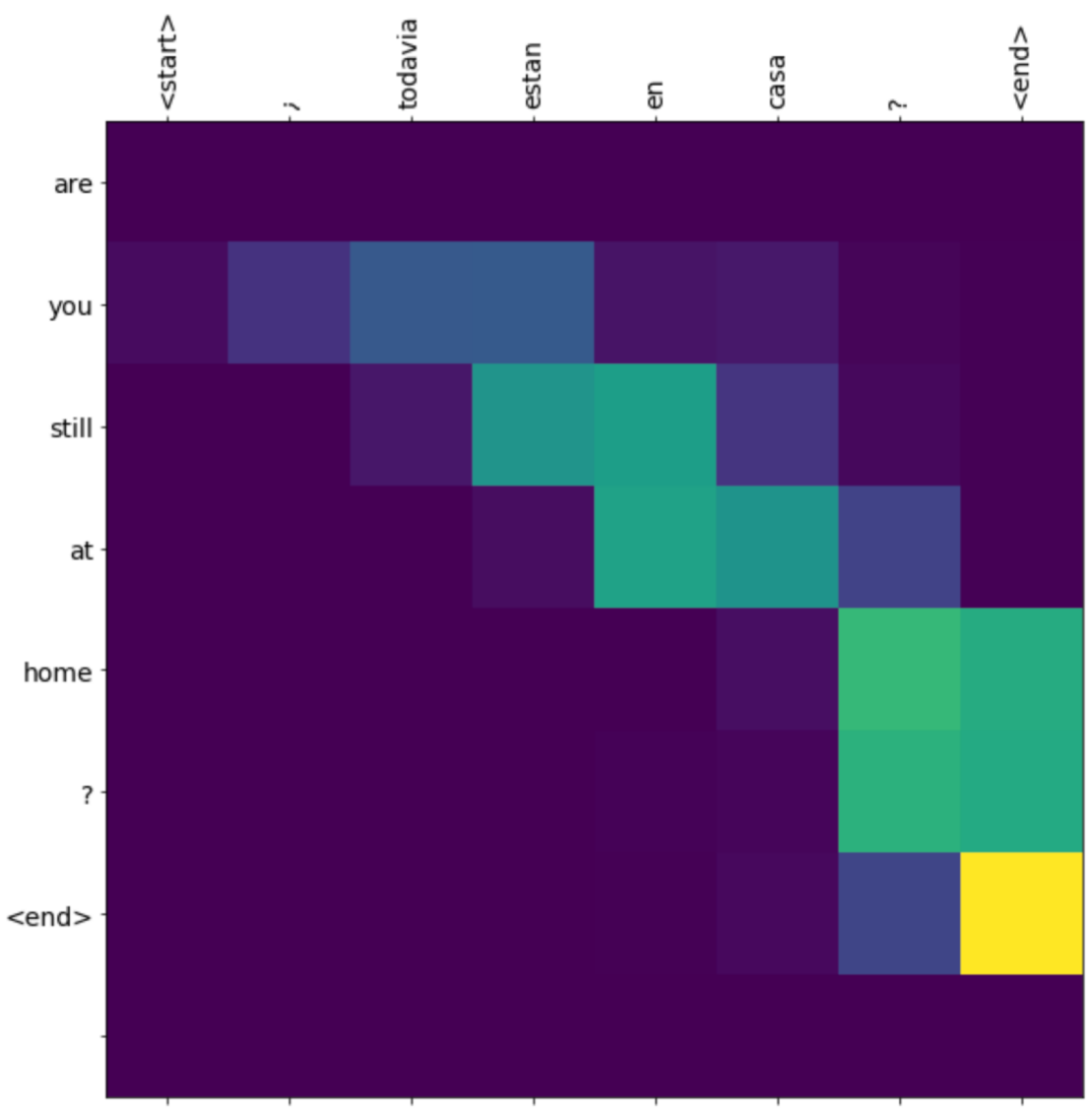


Monotonic alignment (local-m): $p_t = t$

Predictive alignment (local-p): $p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W_p h}_t))$

Attention application



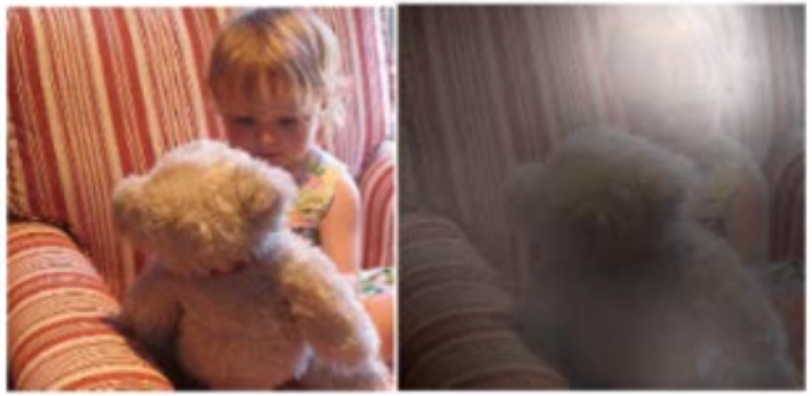


Attention application

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.



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

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

To be continued

Transformer

References

- Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation
- Sequence to Sequence Learning with Neural Networks
- NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE
- Effective Approaches to Attention-based Neural Machine Translation
- Show, Attend and Tell: Neural Image Caption Generation with Visual Attention