Attention Mechanism
Part 1

Zhengyang Wang

Some slides are adapted from Stanford CS224N/Ling284.
(Abigail See and Richard Socher)

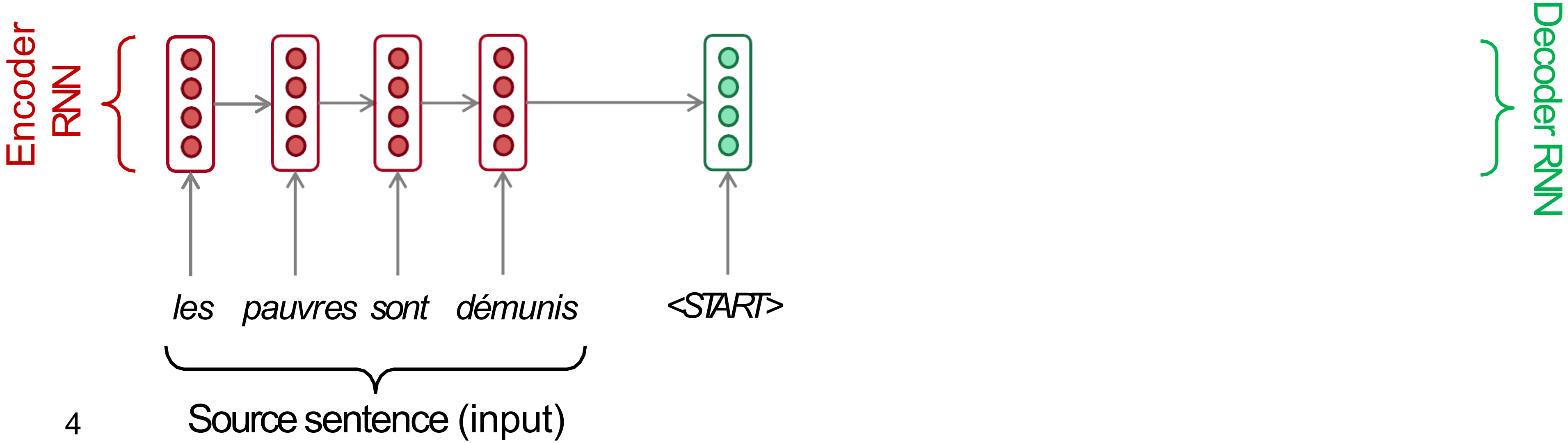
Outlines

- Review the attention mechanism in seq2seq models
- Definition of what the attention mechanism does
- How to use the attention mechanism in general

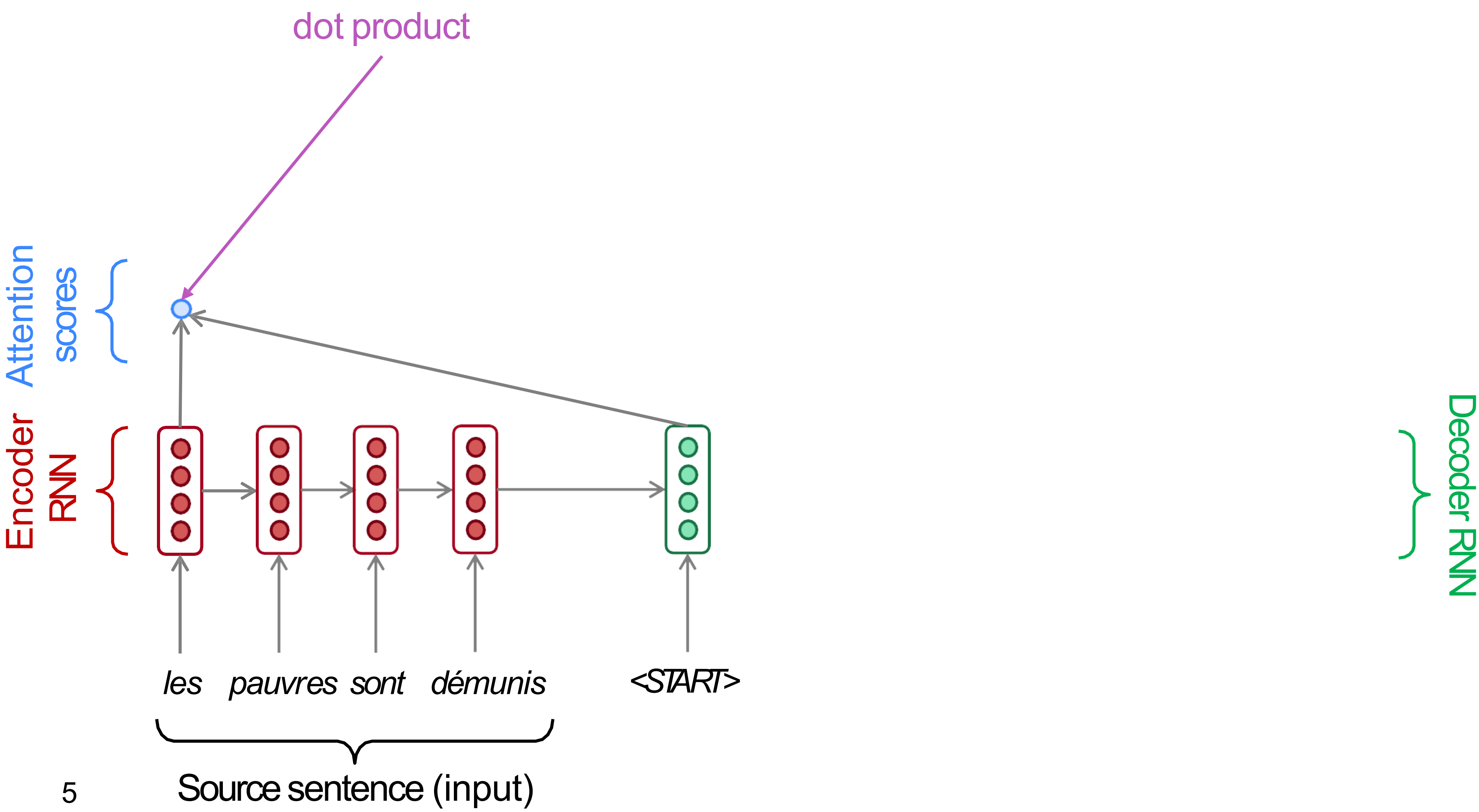
Outlines

- Review the attention mechanism in seq2seq models
- Definition of what the attention mechanism does
- How to use the attention mechanism in general

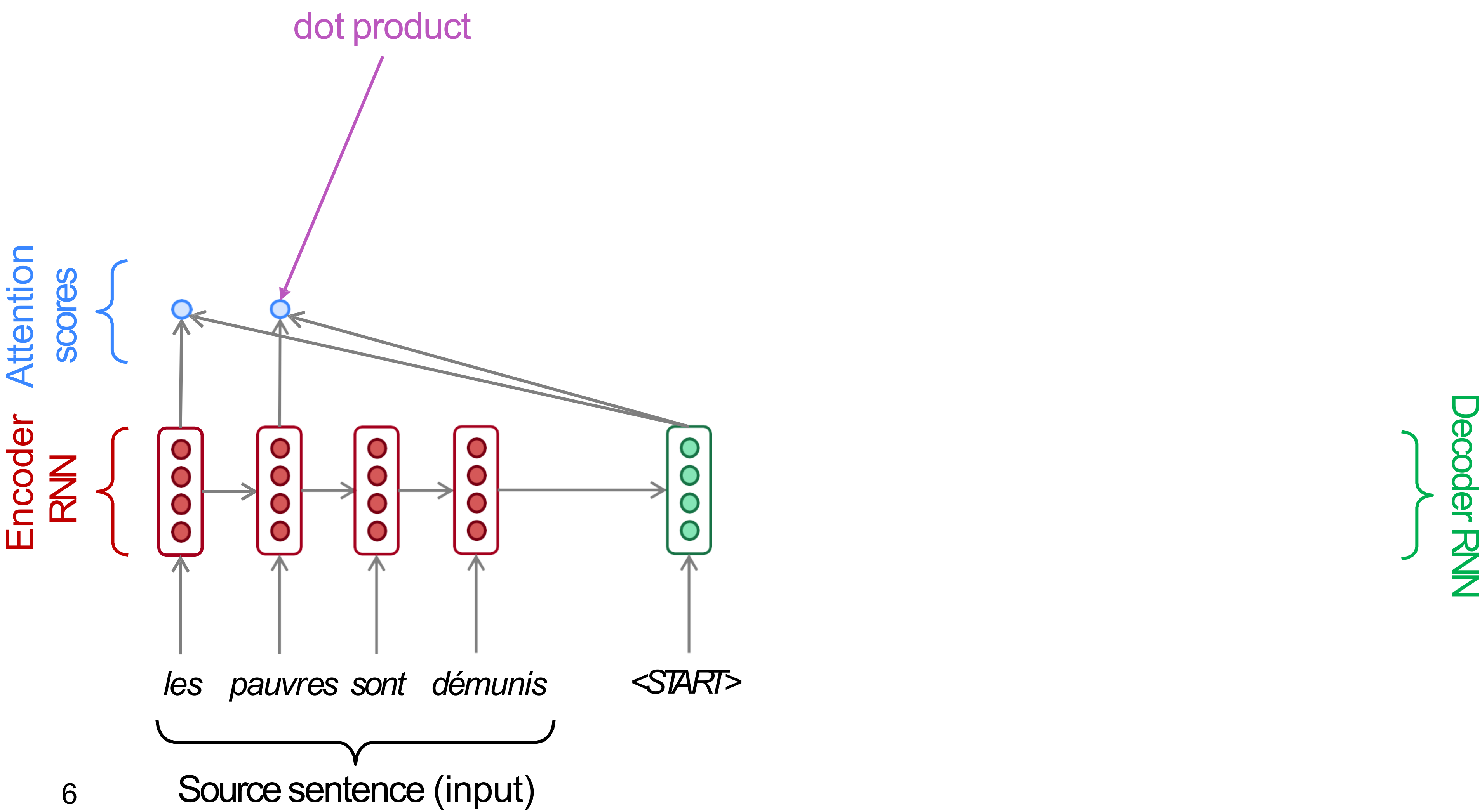
Sequence-to-sequence with attention



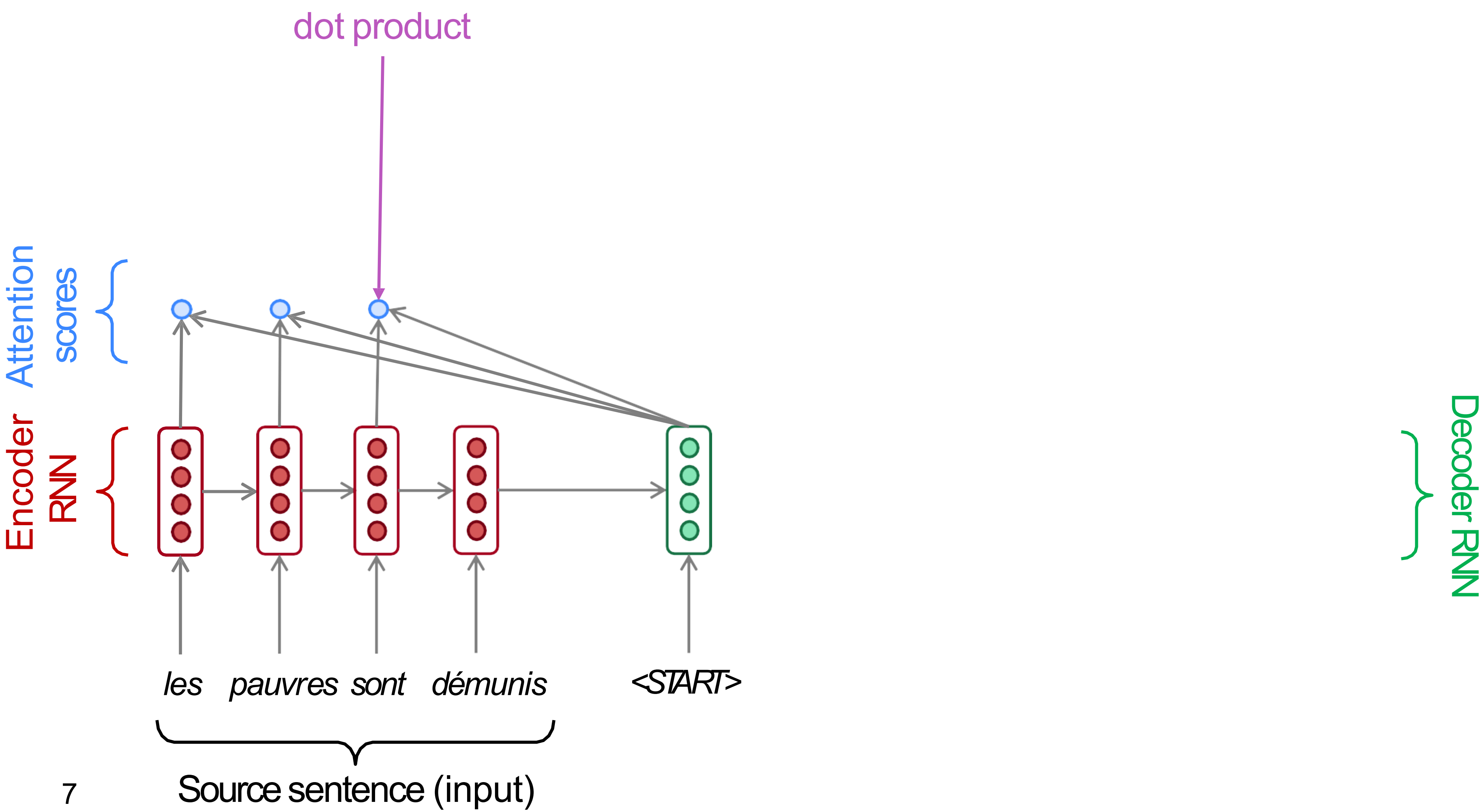
Sequence-to-sequence with attention



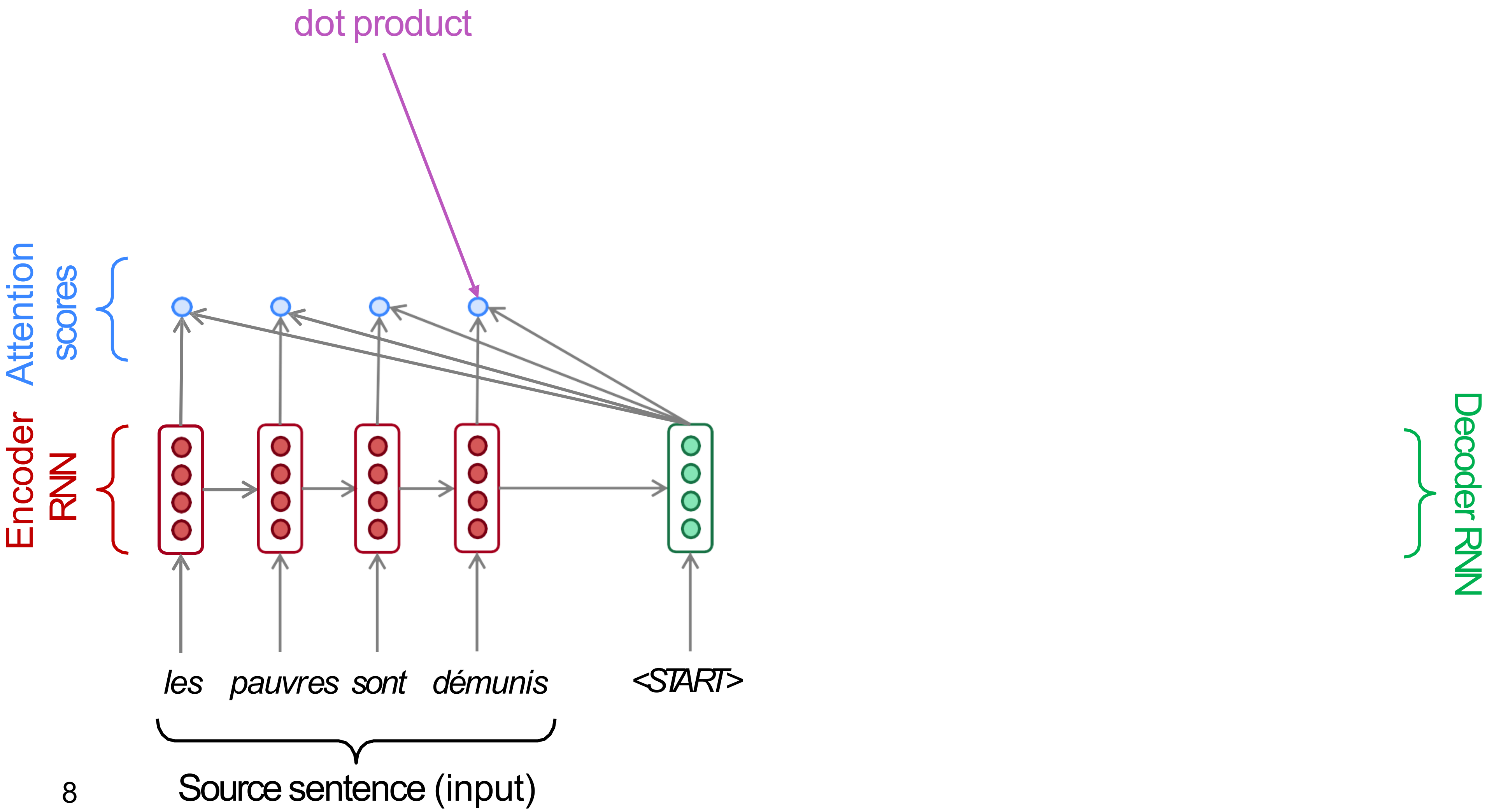
Sequence-to-sequence with attention



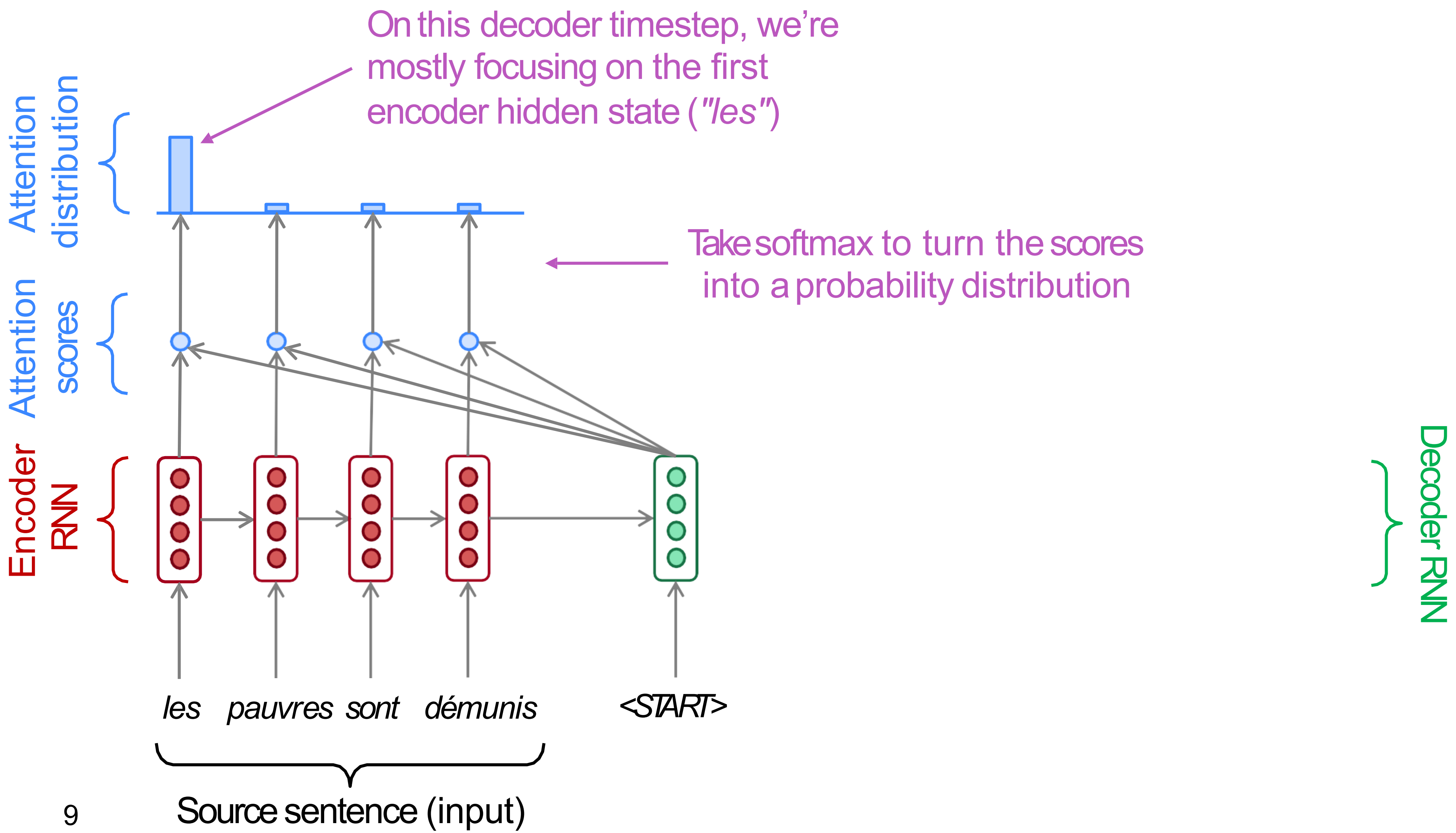
Sequence-to-sequence with attention



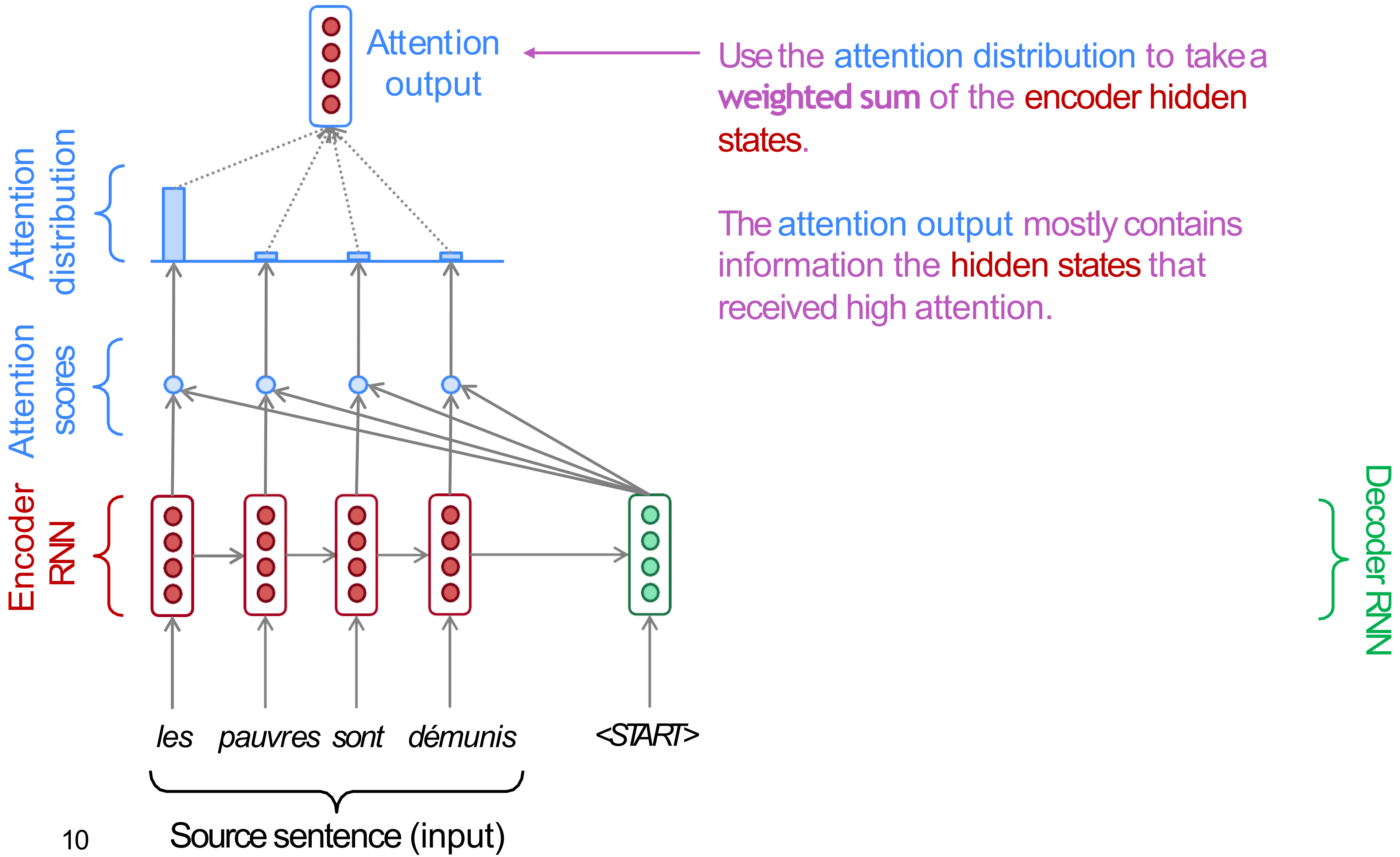
Sequence-to-sequence with attention



Sequence-to-sequence with attention



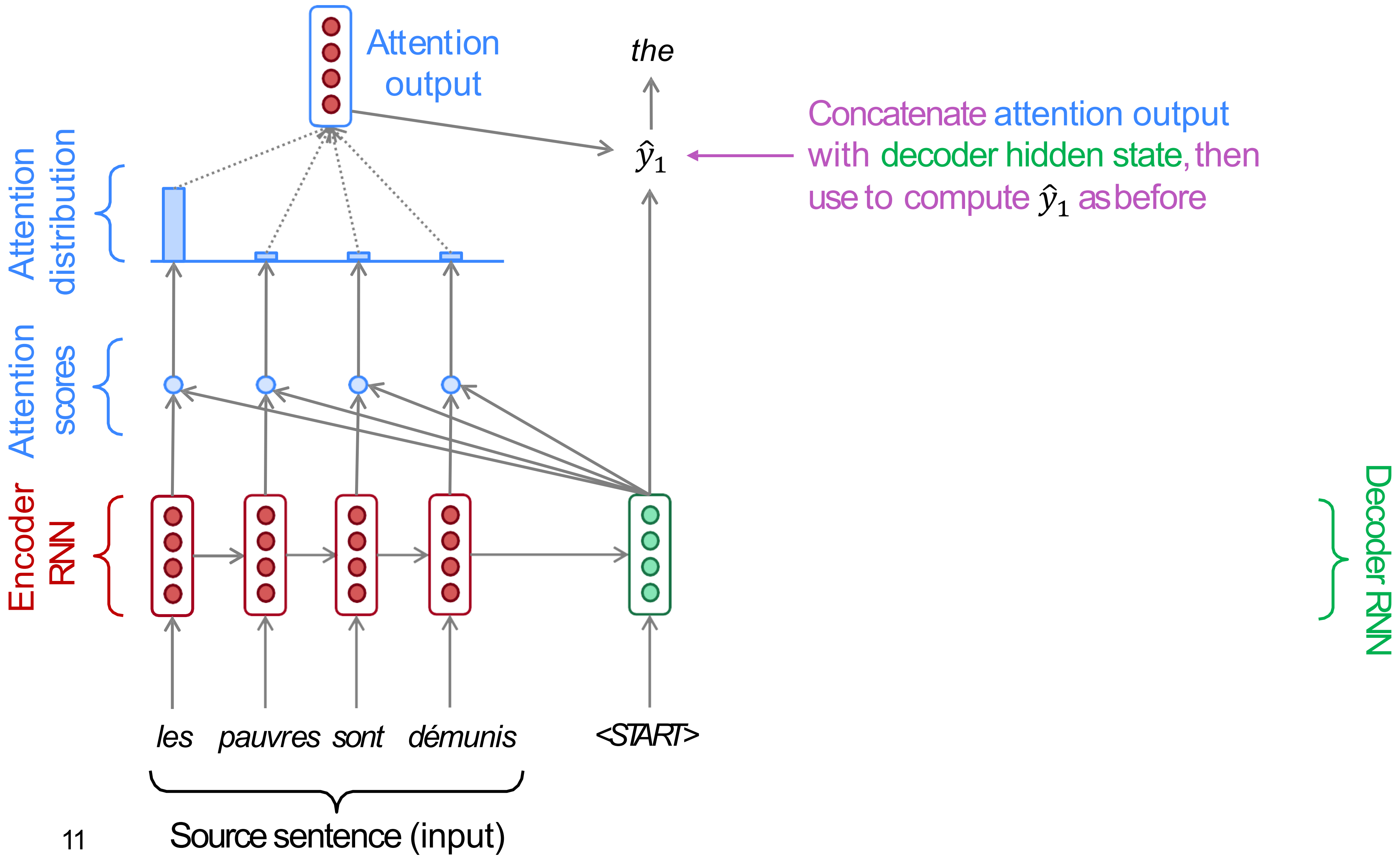
Sequence-to-sequence with attention



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

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

Sequence-to-sequence with attention



Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Attention: in equations

- 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 $\boldsymbol{e}^t$ 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)

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

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

- 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

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

Outlines

- Review the attention mechanism in seq2seq models
- Definition of what the attention mechanism does
- How to use the attention mechanism in general

Attention: in equations

- We have $\boxed{\text{encoder hidden states } h_1, \ldots, h_N \in \mathbb{R}^h}$ **Input 1**
- On timestep t, we have $\boxed{\text{decoder hidden state } s_t \in \mathbb{R}^h}$ **Input 2**
- We get the attention scores $\boldsymbol{e}^t$ 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)

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

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

- 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: in equations

- We have $\boxed{\text{encoder hidden states } h_1, \ldots, h_N \in \mathbb{R}^h}$ **Input 1**
- On timestep *t*, we have $\boxed{\text{decoder hidden state } s_t \in \mathbb{R}^h}$ **Input 2**
- We get the attention scores $\boldsymbol{e}^t$ for this step:

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

Compute attention scores:
<u>dot product</u>

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

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

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

- 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: in equations

- We have $\boxed{\text{encoder hidden states } h_1, \ldots, h_N \in \mathbb{R}^h}$ **Input 1**
- On timestep t, we have $\boxed{\text{decoder hidden state } s_t \in \mathbb{R}^h}$ **Input 2**
- We get the attention scores $\boldsymbol{e}^t$ for this step:

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

Compute attention scores: dot product

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

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

Normalize attention scores: softmax

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

- 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: in equations

- We have $\boxed{\text{encoder hidden states } h_1, \ldots, h_N \in \mathbb{R}^h}$ **Input 1**
- On timestep t, we have $\boxed{\text{decoder hidden state } s_t \in \mathbb{R}^h}$ **Input 2**
- We get the attention scores $\boldsymbol{e}^t$ for this step:

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

Compute attention scores:
dot product

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

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

Normalize attention scores:
softmax

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

Output

- 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: in equations

- We have $\boxed{\text{encoder hidden states } h_1, \ldots, h_N \in \mathbb{R}^h}$ **Input 1**
- On timestep *t*, we have $\boxed{\text{decoder hidden state } s_t \in \mathbb{R}^h}$ **Input 2**
- We get the attention scores $\boldsymbol{e}^t$ for this step:

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

Compute attention scores: <u>dot product</u>

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

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

Normalize attention scores: <u>softmax</u>

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

$$\boxed{\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \underbrace{\boxed{\boldsymbol{h}_i}}_{\text{Input 3}} \in \mathbb{R}^h}$$

Output: sum of Input 3 weighted by normalized attention scores

- 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: in equations

- We have encoder hidden states $h_1, \ldots, h_N \in \mathbb{R}^h$

~~Input 1~~ Key vectors

- On timestep t, we have decoder hidden state $s_t \in \mathbb{R}^h$

~~Input 2~~ Query vector(s)

- We get the attention scores $\boldsymbol{e}^t$ for this step:

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

Compute attention scores: dot product

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

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

Normalize attention scores: softmax

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output $\boldsymbol{a}_t$

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

Value vectors ~~Input 3~~

Output: sum of Input 3 weighted by normalized attention scores

- 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: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors
- Computation steps:
  - Compute attention scores
  - Normalize attention scores
- Output:
  - Sum of value vectors weighted by normalized attention scores

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

We have $n_q = 1$ in the seq2seq example.
We stick to $n_q = 1$ for now.

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

- Computation steps:
  - Compute attention scores: compute the dot product between Q $(n_q = 1)$ and each column in K.

$$\boldsymbol{A} = \boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{K} \in \mathbb{R}^{\boldsymbol{n_q} \times \boldsymbol{n_k}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

- Computation steps:
  - Compute attention scores: compute the dot product between Q $(n_q = 1)$ and each column in K.

$$\boldsymbol{A} = \boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{K} \in \mathbb{R}^{\boldsymbol{n_q} \times \boldsymbol{n_k}}$$

  - Does it put any constraint on the first dimension of Q and K ?

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

- Computation steps:
  - Compute attention scores: compute the dot product between Q $(n_q = 1)$ and each column in K.

$$\boldsymbol{A} = \boldsymbol{Q}^{\boldsymbol{T}}\boldsymbol{K} \in \mathbb{R}^{\boldsymbol{n_q} \times \boldsymbol{n_k}}$$

  - Does it put any constraint on the first dimension of Q and K ?

$$\boldsymbol{d_q} = \boldsymbol{d_k}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$$\boldsymbol{d_q = d_k}$$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: Softmax

$$\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

<u>Constraints:</u>

$\boldsymbol{d_q = d_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: Softmax

$$\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$$

  - If $n_q \neq 1$, should we perform Softmax over column or row?

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors
- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: Softmax

$$\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$$

  - If $n_q \neq 1$, should we perform Softmax over column or row?

Row

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

  - Does it put any constraint on the second dimension of V and K ?

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

  - Does it put any constraint on the second dimension of V and K ?

$$\boldsymbol{n_v = n_k}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

$\boldsymbol{n_v = n_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boldsymbol{d_v \times n_q}}$$

Attention: definition

- Three inputs:
  - Query vector(s)
  - Key vectors
  - Value vectors

- Three input matrices:
  - $Q \in \mathbb{R}^{d_q \times n_q}$
  - $K \in \mathbb{R}^{d_k \times n_k}$
  - $V \in \mathbb{R}^{d_v \times n_v}$

Constraints:

$\boldsymbol{d_q = d_k}$

$\boldsymbol{n_v = n_k}$

- Computation steps:
  - Compute attention scores: $\boldsymbol{A = Q^T K} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$
  - Normalize attention scores: $\boldsymbol{A = Softmax(A)} \in \mathbb{R}^{\boldsymbol{n_q \times n_k}}$

- Output:
  - Sum of value vectors weighted by normalized attention scores:

$$\boldsymbol{Output = V \cdot A^T} \in \mathbb{R}^{\boxed{\boldsymbol{d_v \times n_q}}}$$

  - The shape of the output is determined by the number of query vectors and the dimension of value vectors.

Outlines

- Review the attention mechanism in seq2seq models
- Definition of what the attention mechanism does
- How to use the attention mechanism in general

Use Attention

- Use attention:
  - Determine Q, K, V
    - In the seq2seq example, we have $K = V$ (encoder hidden states) and a single query vector $Q = q$ (the decoder hidden state at time *t*).

Use Attention

- Use attention:
  - Determine Q, K, V
    - In the seq2seq example, we have $K = V$ (encoder hidden states) and a single query vector $Q = q$ (the decoder hidden state at time *t*).
  - Define the way to compute and normalize attention scores
    - Dot product + Softmax is just one way.
    - Check paper "Non-local Neural Networks" for other ways.
    - For example: dot product + $1/n_k$; use a neural network.

Use Attention

- Can we use attention just as convolution or fully-connected layer?
  - One input X
  - Some training parameters
  - One output Y

- How can we have Q, K, V if we only have one input X ?

Use Attention

- Can we use attention just as convolution or fully-connected layer?
  - One input X
  - Some training parameters
  - One output Y

- How can we have Q, K, V if we only have one input X ?

- Self-Attention
  - $Q = K = V = X$
  - Any problem?

Use Attention

- Can we use attention just as convolution or fully-connected layer?
  - One input X
  - Some training parameters
  - One output Y
- How can we have Q, K, V if we only have one input X ?
- Self-Attention
  - $Q = K = V = X$
  - Any problem?
    - No training parameter involved.
    - It does not make much sense.

Self-Attention

- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

Self-Attention

- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

- What is the shape of the output Y ?

Self-Attention

- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

- What is the shape of the output Y ?

$$\boldsymbol{d_v \times n_x}$$

Self-Attention

- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

- What is the shape of the output Y ?

$$\boldsymbol{d_v \times n_x}$$

- **If we omit $\boldsymbol{W_v}$ and simply have $\boldsymbol{V = X}$, each column in $\boldsymbol{Y}$ is a weighted sum of all column vectors in $\boldsymbol{X}$.**

Self-Attention

- **If we omit W_v and simply have $V = X$, each column in Y is a weighted sum of all column vectors in X.**

- Comparison with convolution and fully-connected layer

Self-Attention

- **If we omit W_v and simply have $V = X$, each column in $\boldsymbol{Y}$ is a weighted sum of all column vectors in X.**

- Comparison with convolution and fully-connected layer
  - Convolution
    - Each column in Y is computed from column vectors within a local range in X.

Self-Attention

- **If we omit W_v and simply have $V = X$, each column in Y is a weighted sum of all column vectors in X.**

- Comparison with convolution and fully-connected layer
  - Convolution
    - Each column in Y is computed from column vectors within a local range in X.
  - Fully-connected layer
    - The weights in the weighted sum is not input-dependent.

Multi-Head Self-Attention

- Instead of having $Q = K = V = X$, use X to generate Q, K, V.
  - Suppose $X \in \mathbb{R}^{d_x \times n_x}$
  - Three independent linear transformations:
    - $Q = W_q X \in \mathbb{R}^{d_q \times n_x}$
    - $K = W_k X \in \mathbb{R}^{d_k \times n_x}$
    - $V = W_v X \in \mathbb{R}^{d_v \times n_x}$
  - $W_q \in \mathbb{R}^{d_q \times d_x}, W_k \in \mathbb{R}^{d_k \times d_x}, W_v \in \mathbb{R}^{d_v \times d_x}$ are trainable parameters.
  - W_q, W_k must satisfy the constraints $d_q = d_k$.

- Do the above process for multiple times <u>independently</u>.
  - Each time results in an output Y_i.
  - Concatenate all the Y_i as the final output.