On the Path to Transformers

Sequence Modeling

Armen Aghajanyan

Limitations of RNNs and LSTMs

1. Processes inputs one at a time → Limited parallelization → slower training and potentially inference

2. Vanishing Gradients → Difficulty in learning long-range dependencies → Information from early time steps gets diluted

3. Limited Context Window → Practical limit on sequence length (typically ~100-200 tokens) → Struggles with very long-term dependencies

4. Computational Complexity → O(n) complexity for sequence length n → Becomes prohibitive for very long sequences

5. Fixed-size Hidden State → Bottleneck for information flow → May not capture all relevant information for complex tasks

Dive: Vanishing/Exploding Gradient

RNN Basics:

- Hidden state update: $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$

- Output: $y_t = W_y h_t + b_y$

Backpropagation Through Time (BPTT):

Gradient of loss L with respect to W_h at time step t:

$$\frac{\partial L}{\partial W_h} = \sum_{k=1}^{t} \frac{\partial L}{\partial y_t} \cdot \frac{\partial y_t}{\partial h_t} \cdot \frac{\partial h_t}{\partial h_k} \cdot \frac{\partial h_k}{\partial W_h}$$

Gradient Calculation:

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} \mathrm{diag}(1 - \tanh^2(W_h h_{i-1} + W_x x_i + b)) \cdot W_h$$

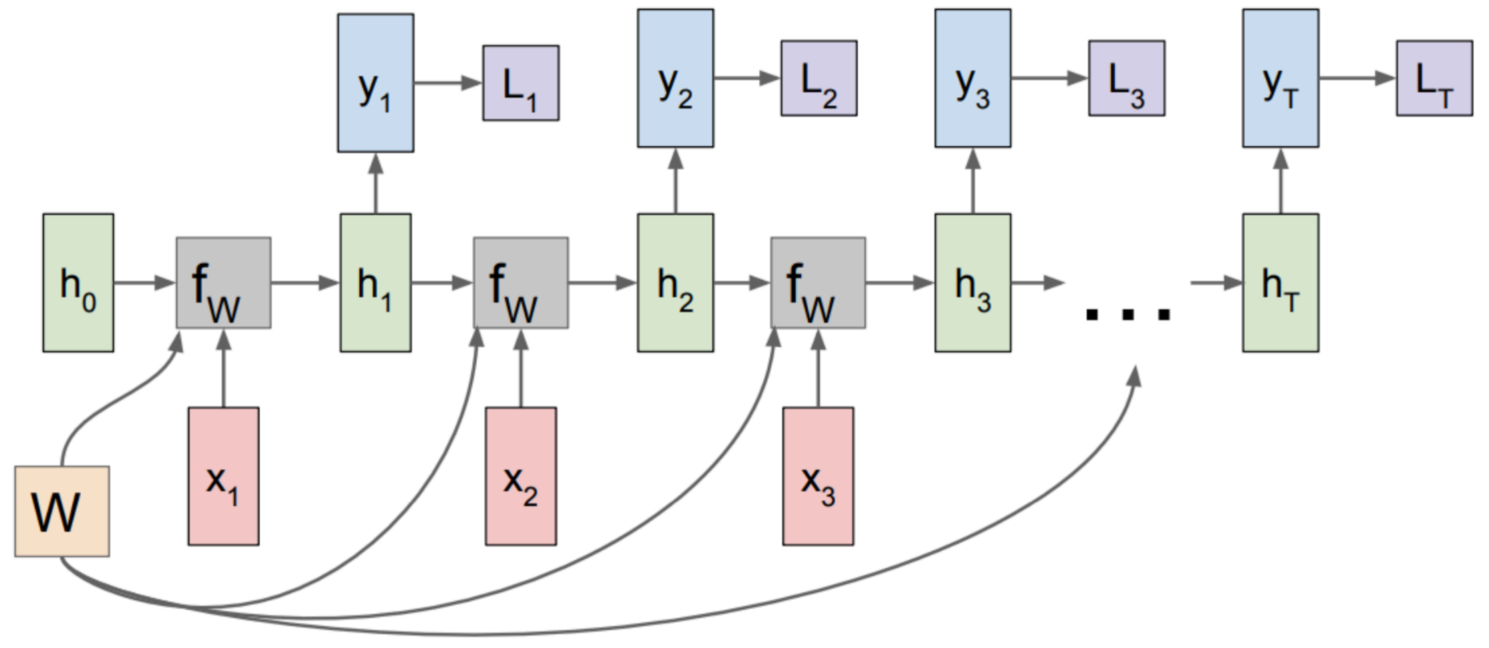
Problem:

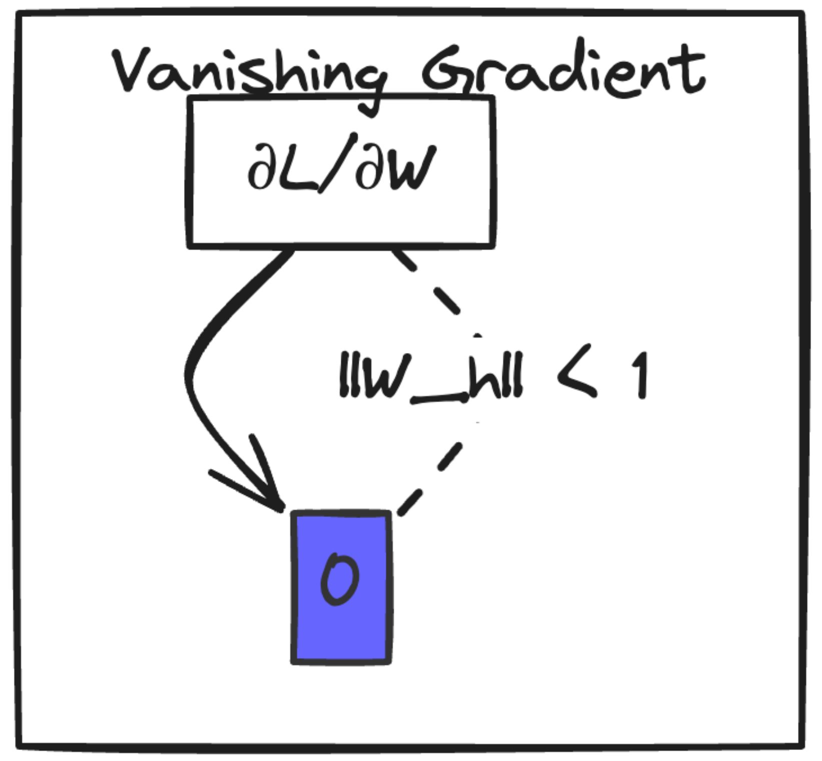
If ||W_h|| > 1: Exploding gradients

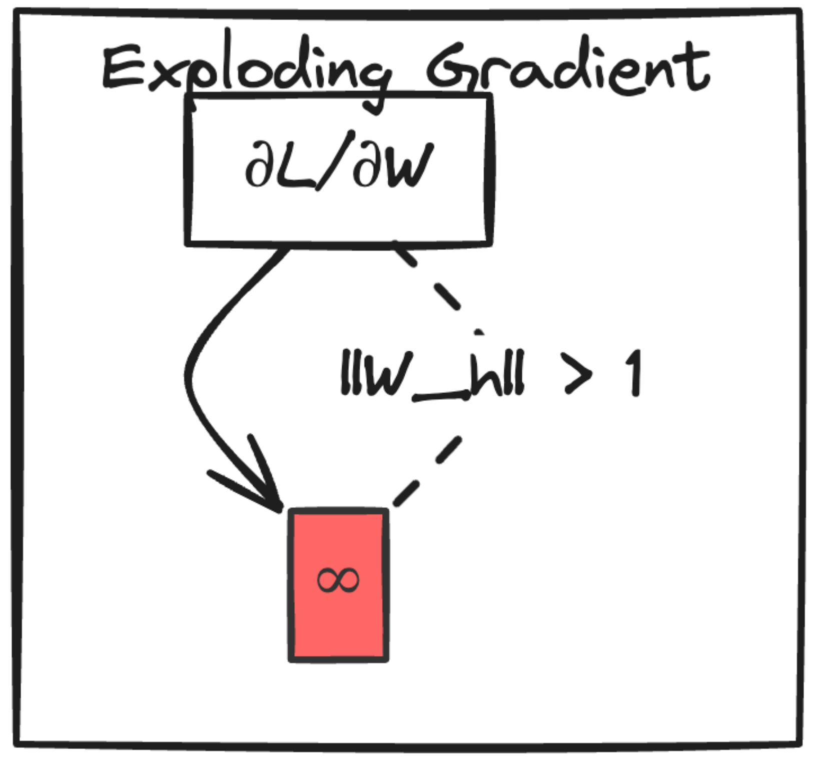
If ||W_h|| < 1: Vanishing gradients

Consequences:

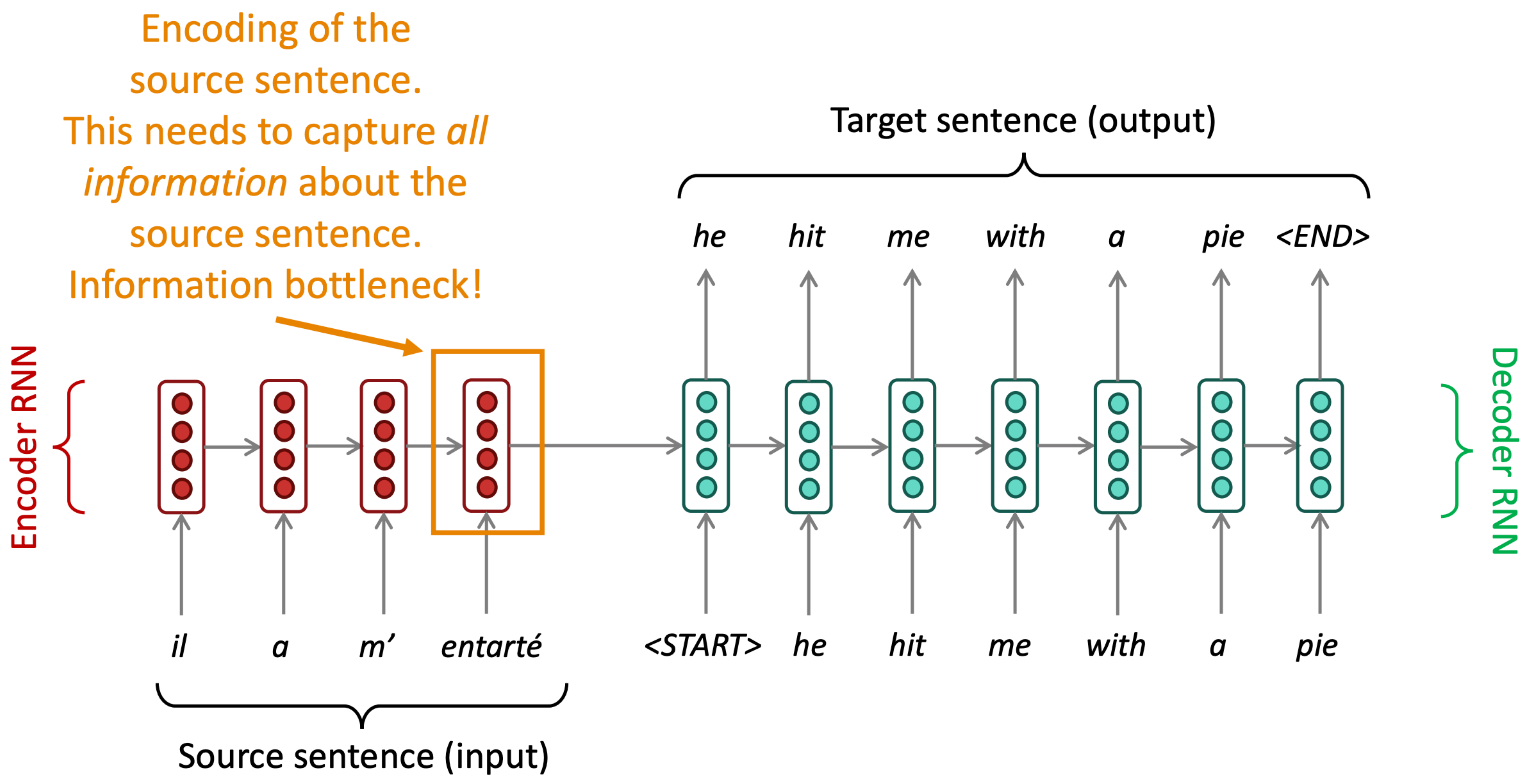
- Exploding: Unstable training, large parameter updates
- Vanishing: Difficulty learning long-term dependencies







Dive: Bottleneck Problem


Encoding of the source sentence. This needs to capture *all information* about the source sentence. Information bottleneck!
Target sentence (output)
he
hit
me
with
a
pie
<END>
Encoder RNN
Decoder RNN
il
a
m'
entarté
<START>
he
hit
me
with
a
pie
Source sentence (input)

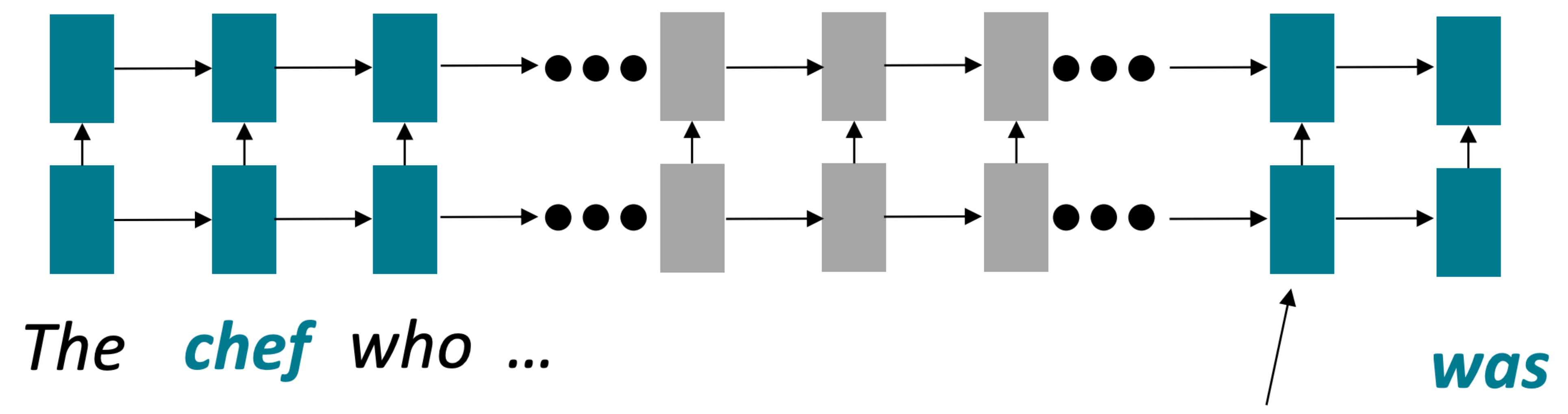
Dive: Linear Interaction Distance

O(sequence length) steps for distant word pairs to interact means:

1. Hard to learn long-distance dependencies (because gradient problems!)

2. Linear order of words is "baked in"; we already know linear order isn't the right way to think about sentences…

(Provocative statement)

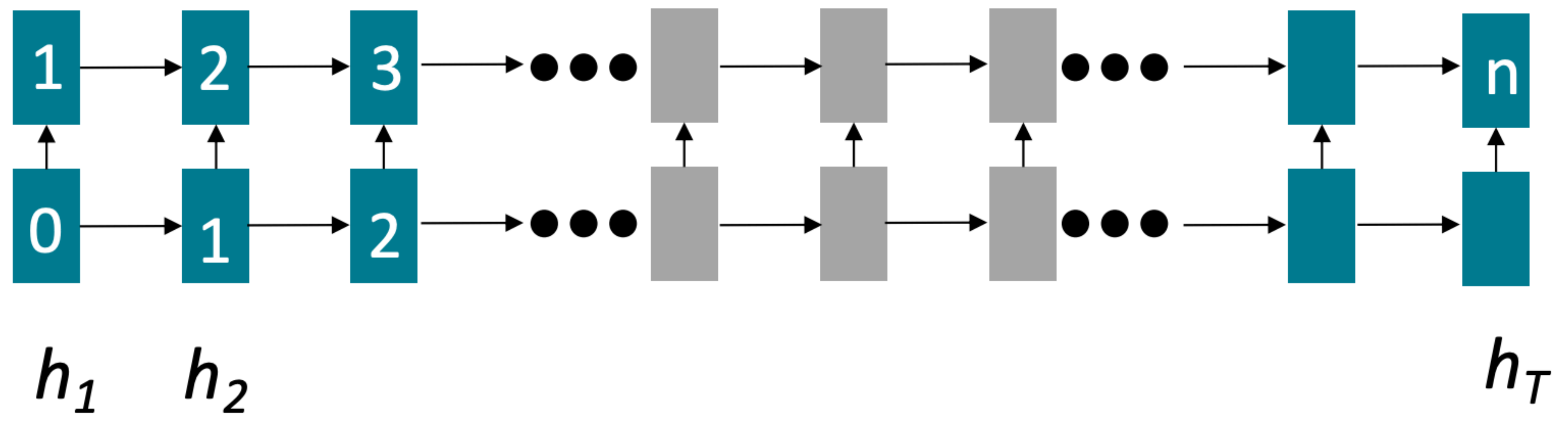


Info of ***chef*** has gone through O(sequence length) many layers!

Dive: Lack of Parallelizability

Forward and backward passes have O(sequence length) unparallelizable operations.

- GPUs can perform a bunch of independent computations at once!
- But future RNN hidden states can't be computed in full before past RNN hidden states have been computed
- Inhibits training on very large datasets!



Numbers indicate min # of steps before a state can be computed

The Need for Better Sequence Models

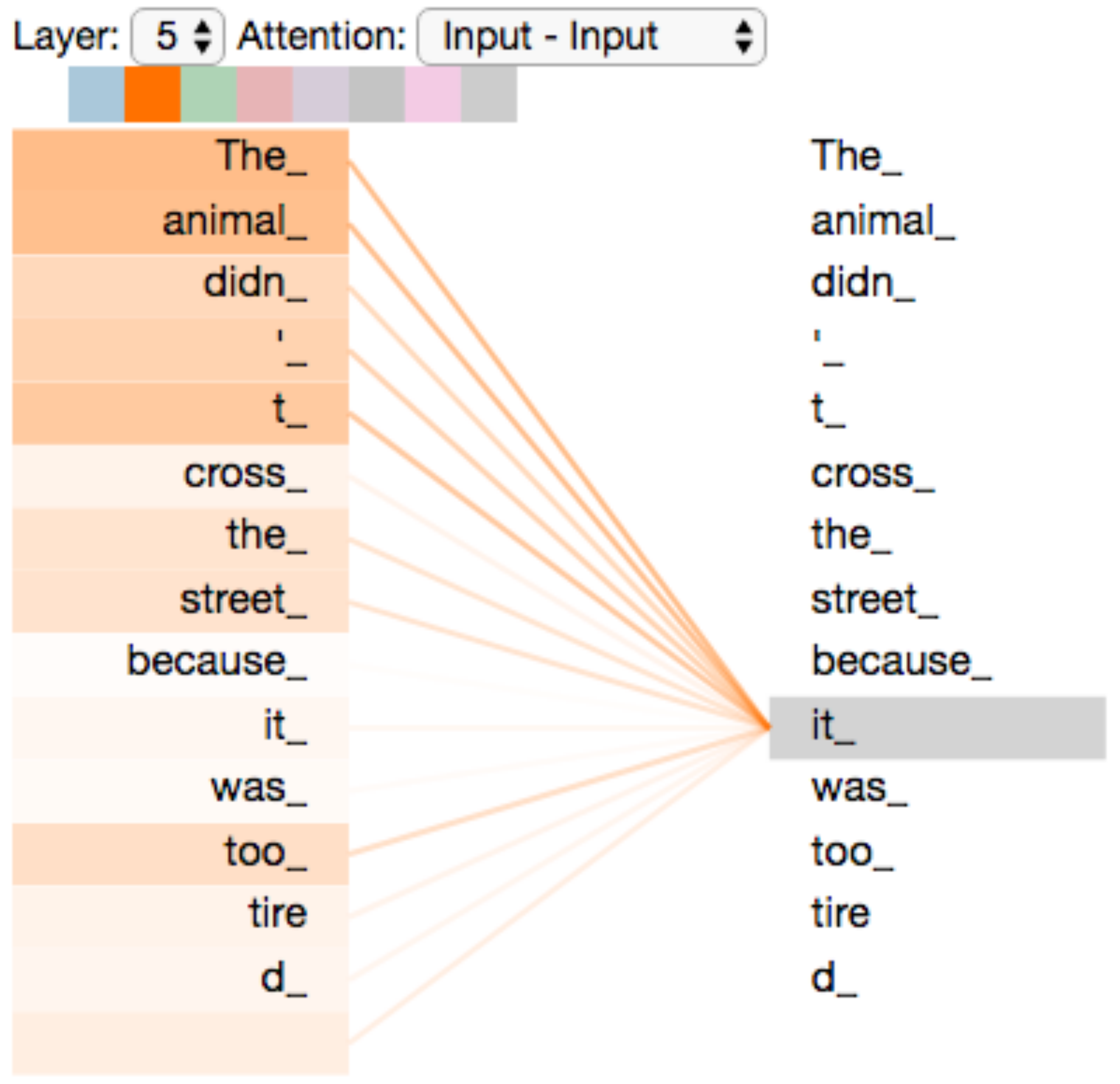
RNNs/LSTMs Limitations
Need for Better Models
Handle Longer Sequences
Capture Complex Dependencies
Parallel Processing

The Need for Better Sequence Models

Information Flow: Attention Mechanism

- Allows to "focus attention" on particular aspects of the input text

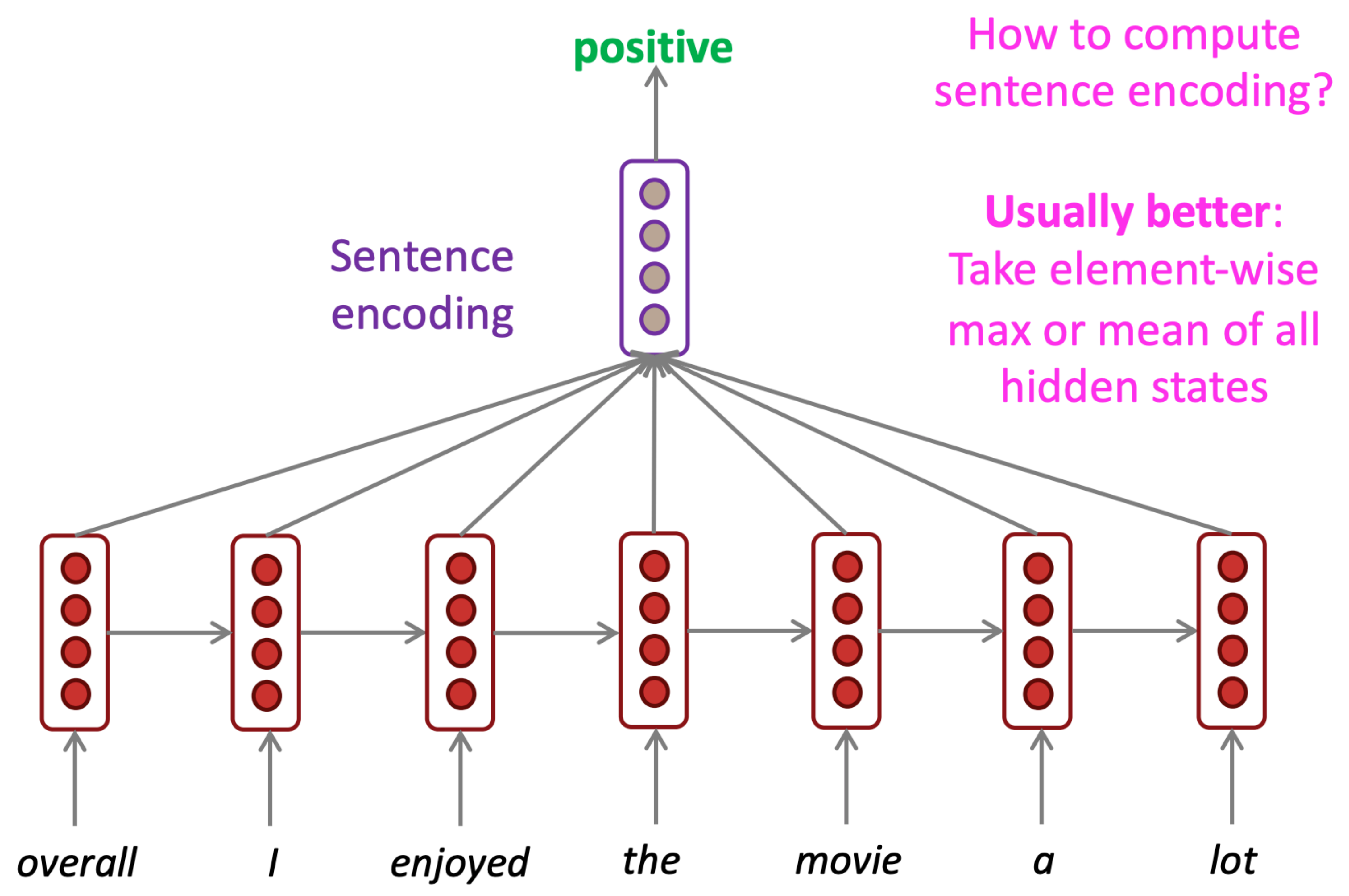
- Core idea: on each step of the decoder, use direct connection to the representations to focus on a particular part of the source sequence



Basic Information Flow

Mean Pooling

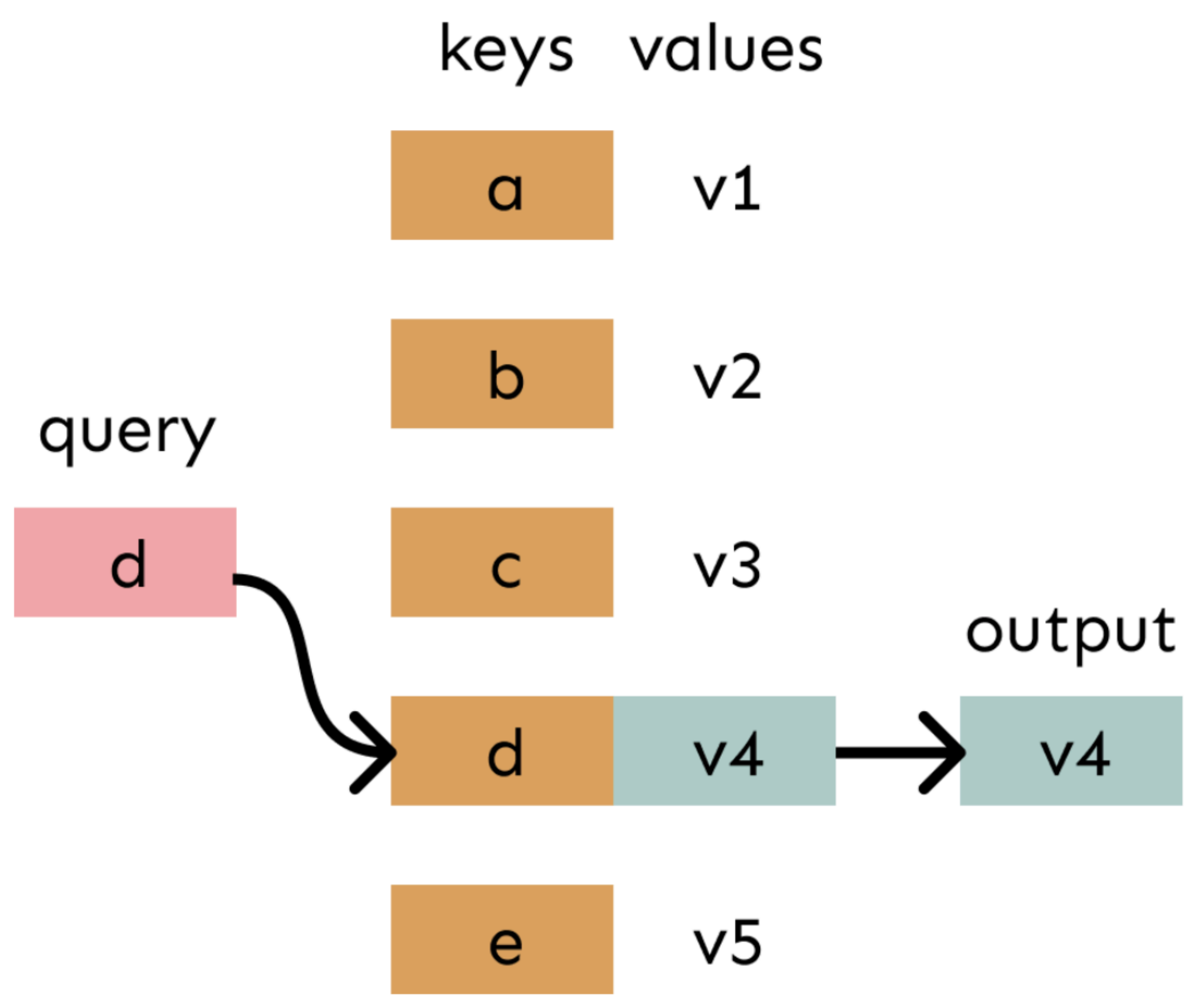
- **Starting point:** a very basic way of 'passing information from the representations' is to average


positive
How to compute sentence encoding?
Sentence encoding
Usually better: Take element-wise max or mean of all hidden states
overall
I
enjoyed
the
movie
a
lot

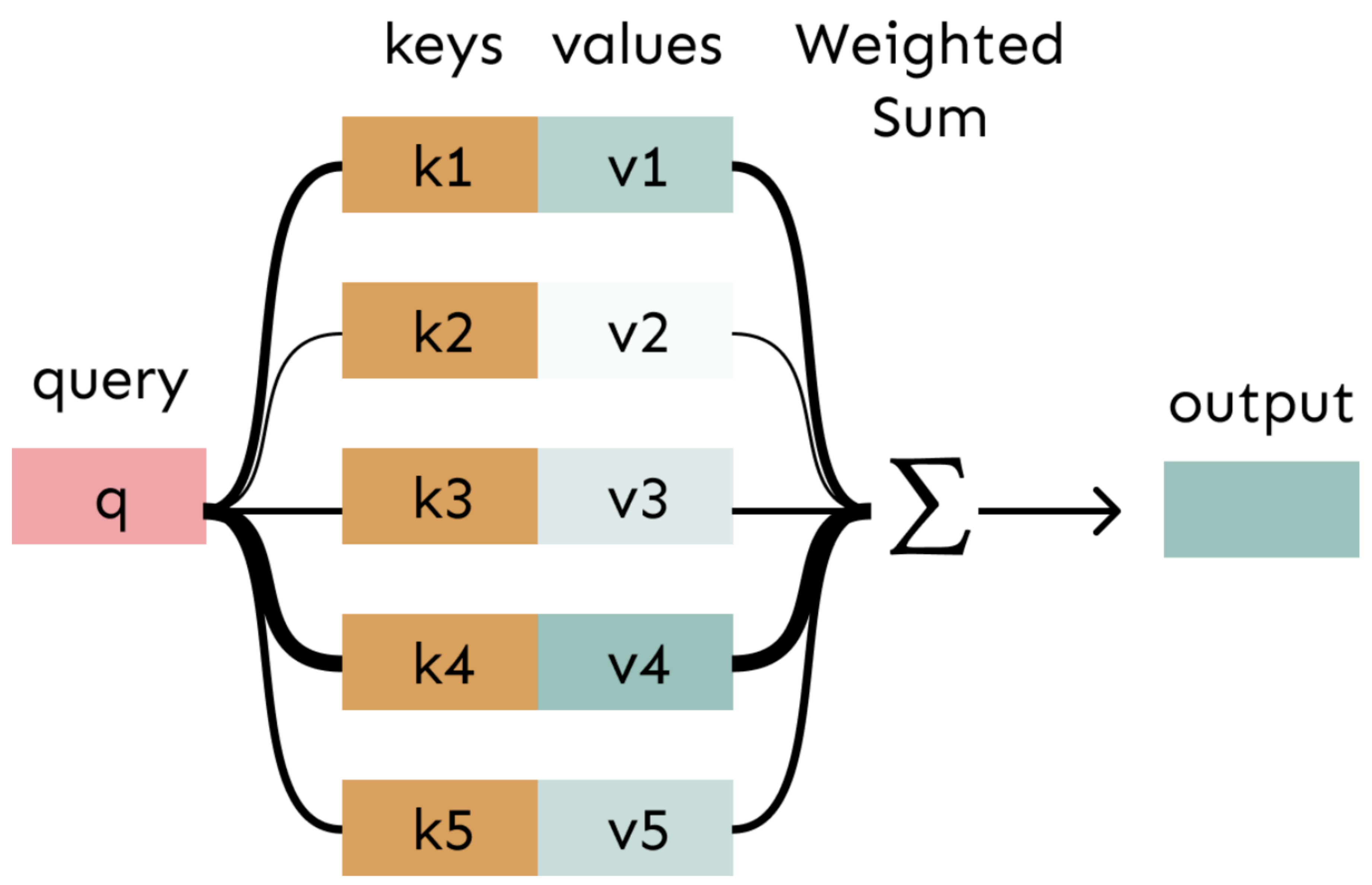
Complex Information Flow

Hard Lookups & Attention

In a lookup table, we have a table of keys that map to values. The query matches one of the keys, returning its value.



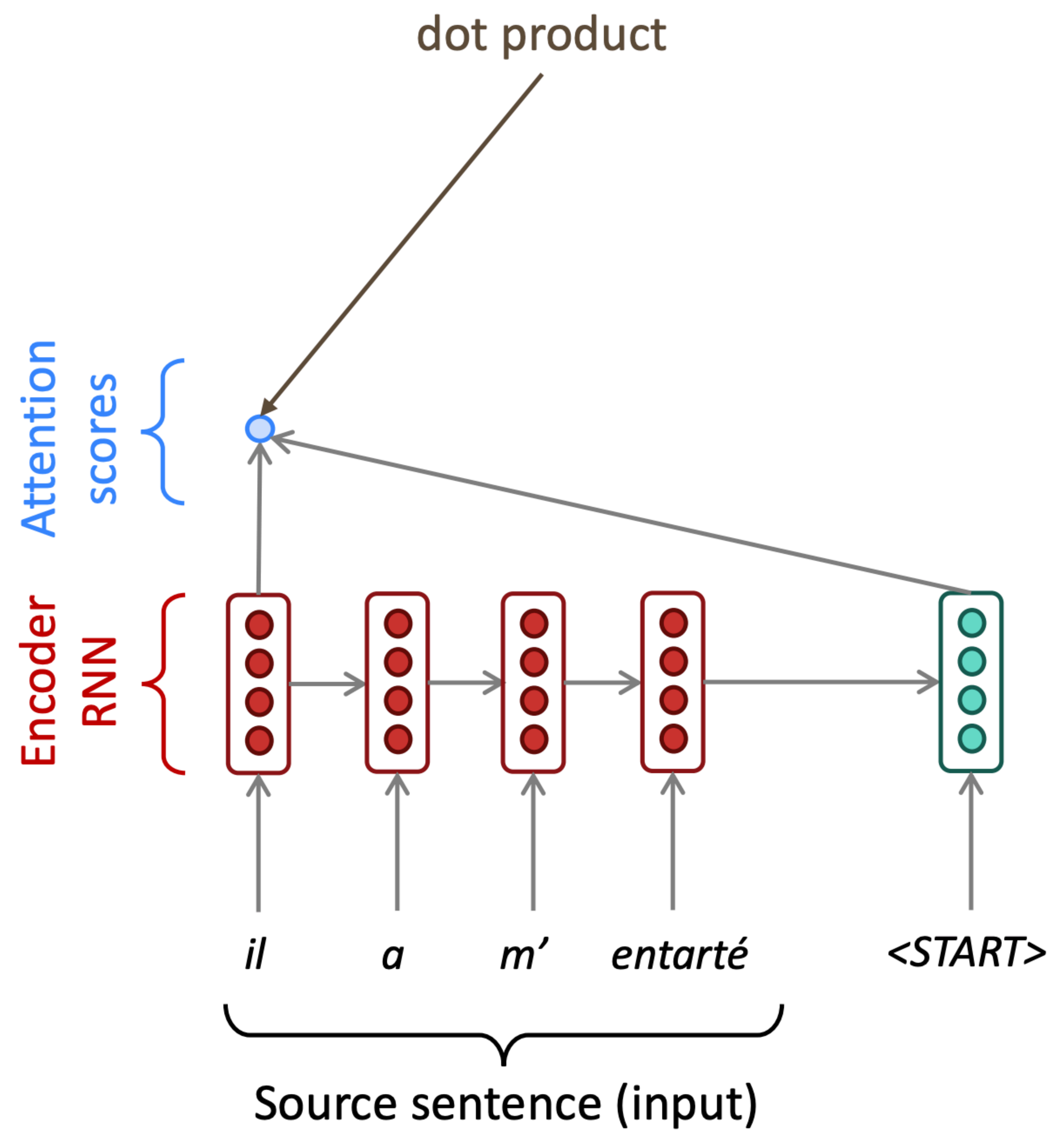
In attention, the query matches all keys softly, to a weight between 0 and 1. The keys' values are multiplied by the weights and summed.



Simple Walkthrough Attention

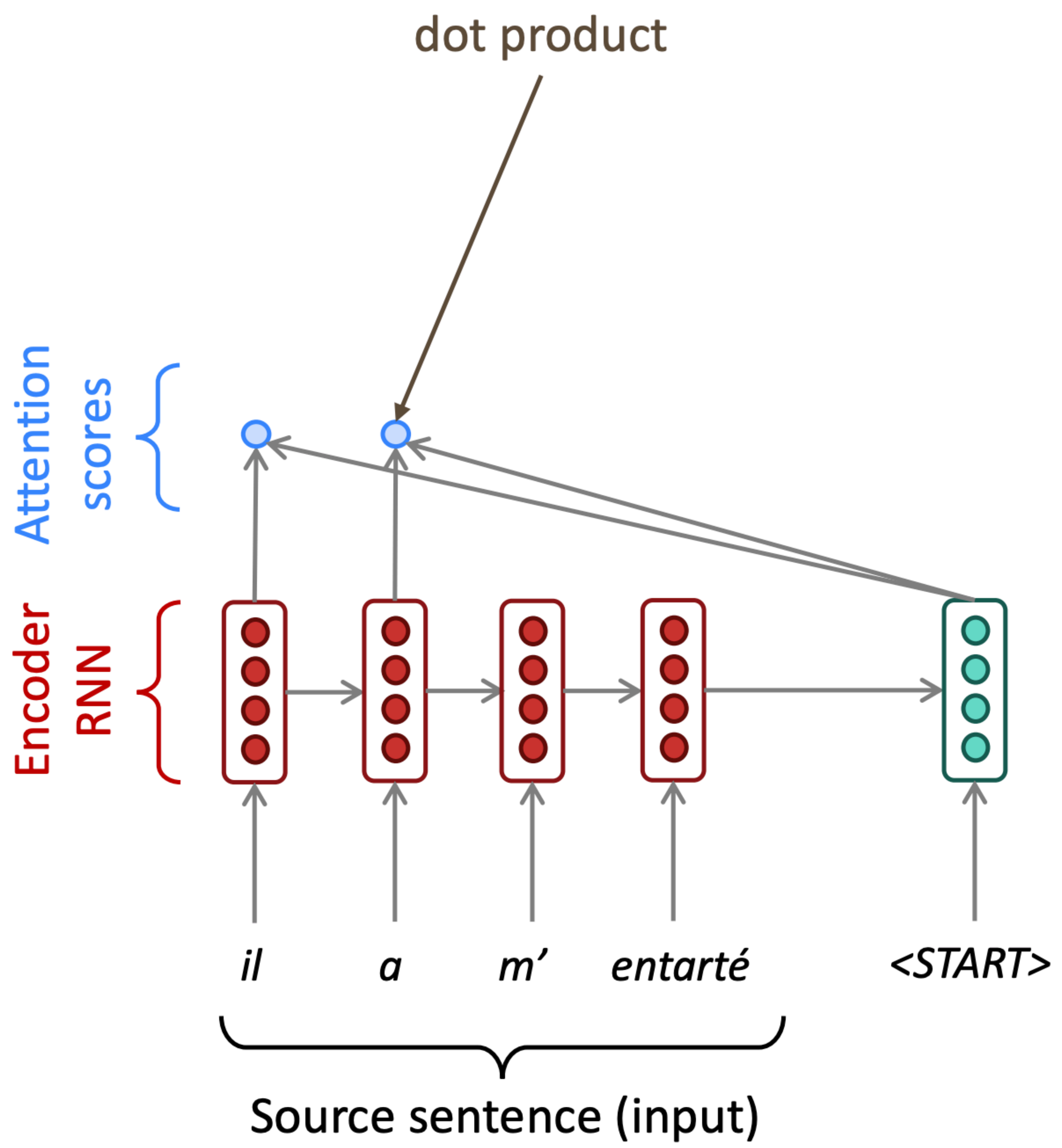
With RNN's

Core idea: on each step of the decoder, use direct connection to the representations to focus on a particular part of the source sequence



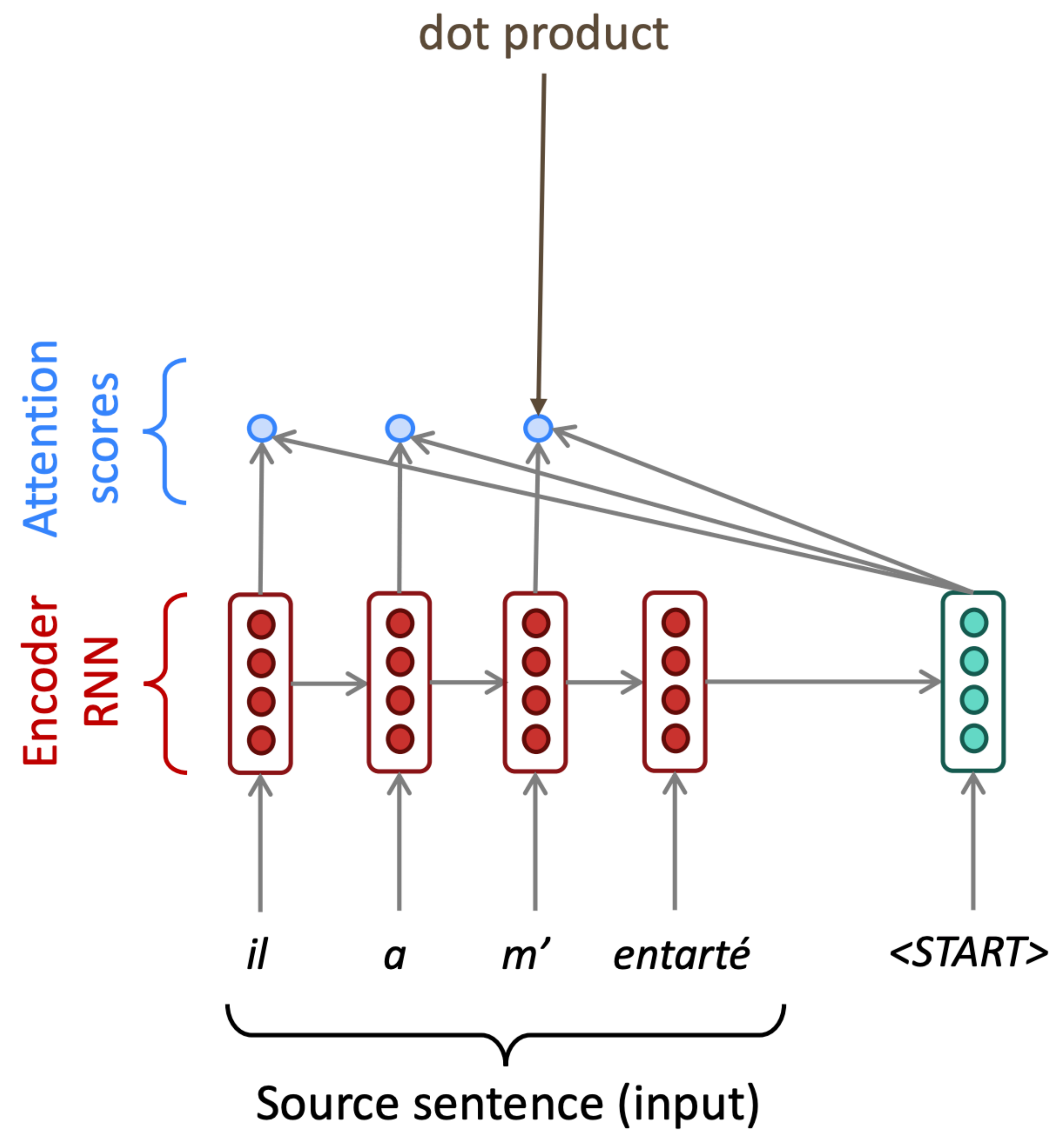
Simple Walkthrough Attention

With RNN's



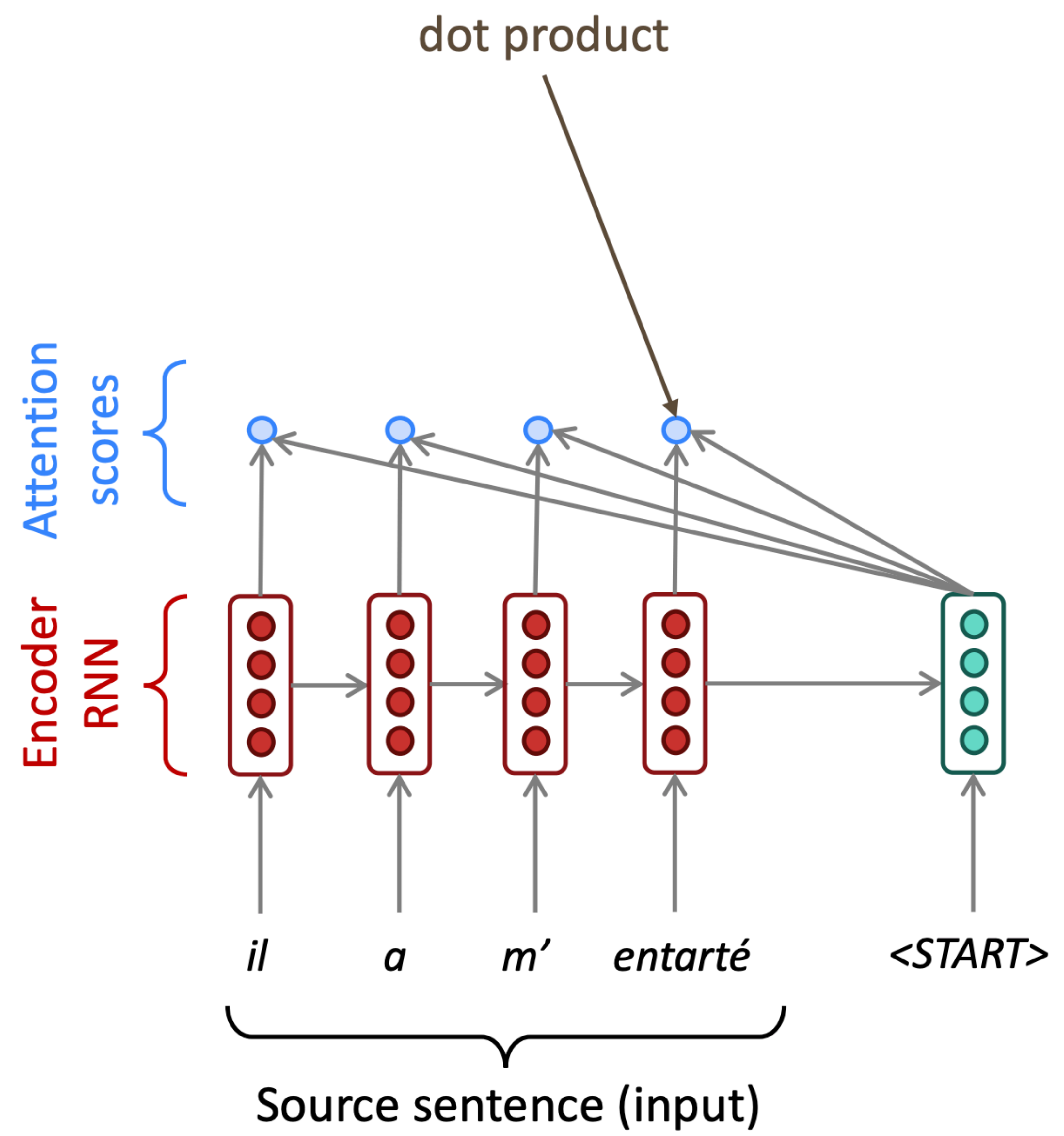
Simple Walkthrough Attention

With RNN's



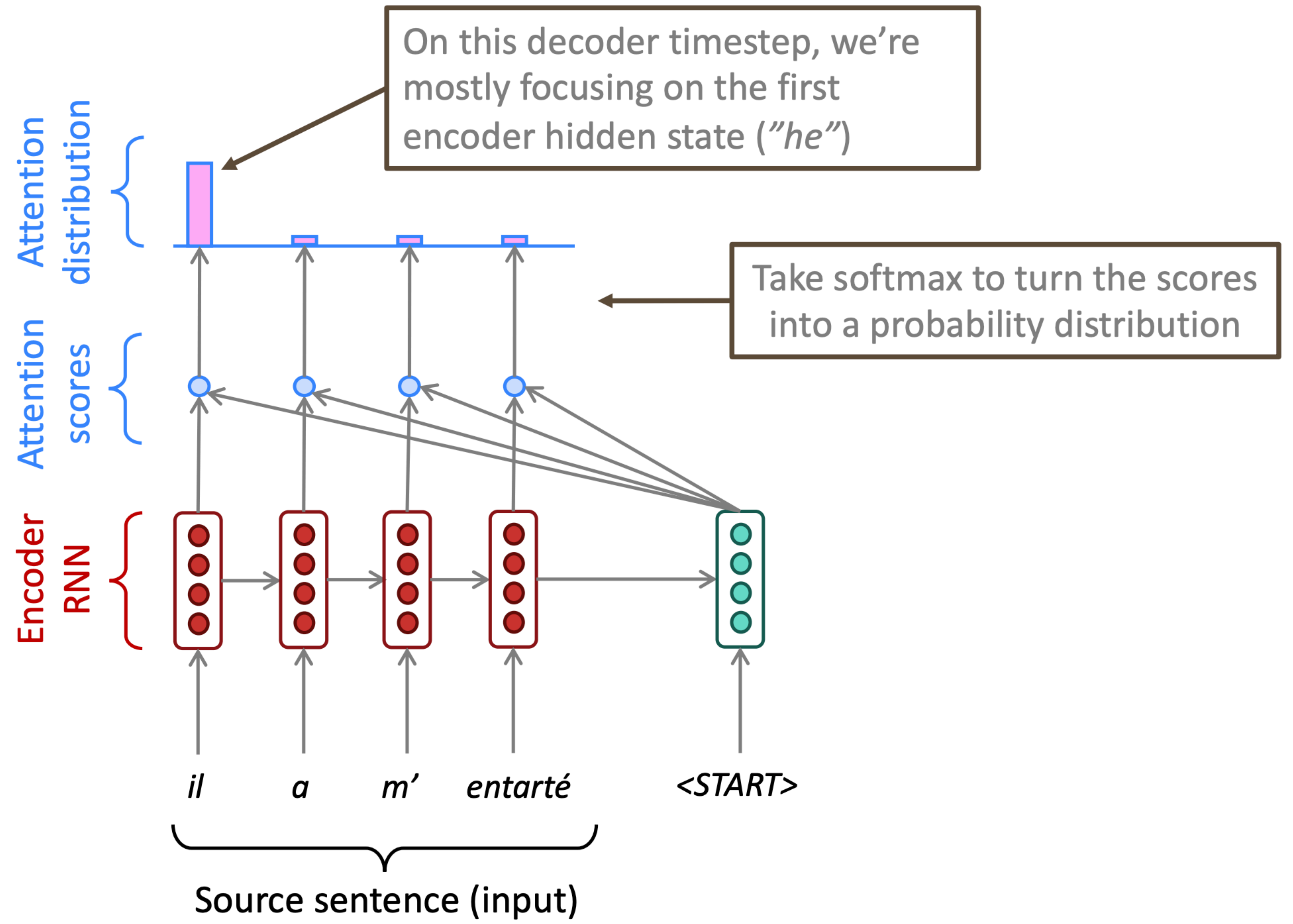
Simple Walkthrough Attention

With RNN's



Simple Walkthrough Attention

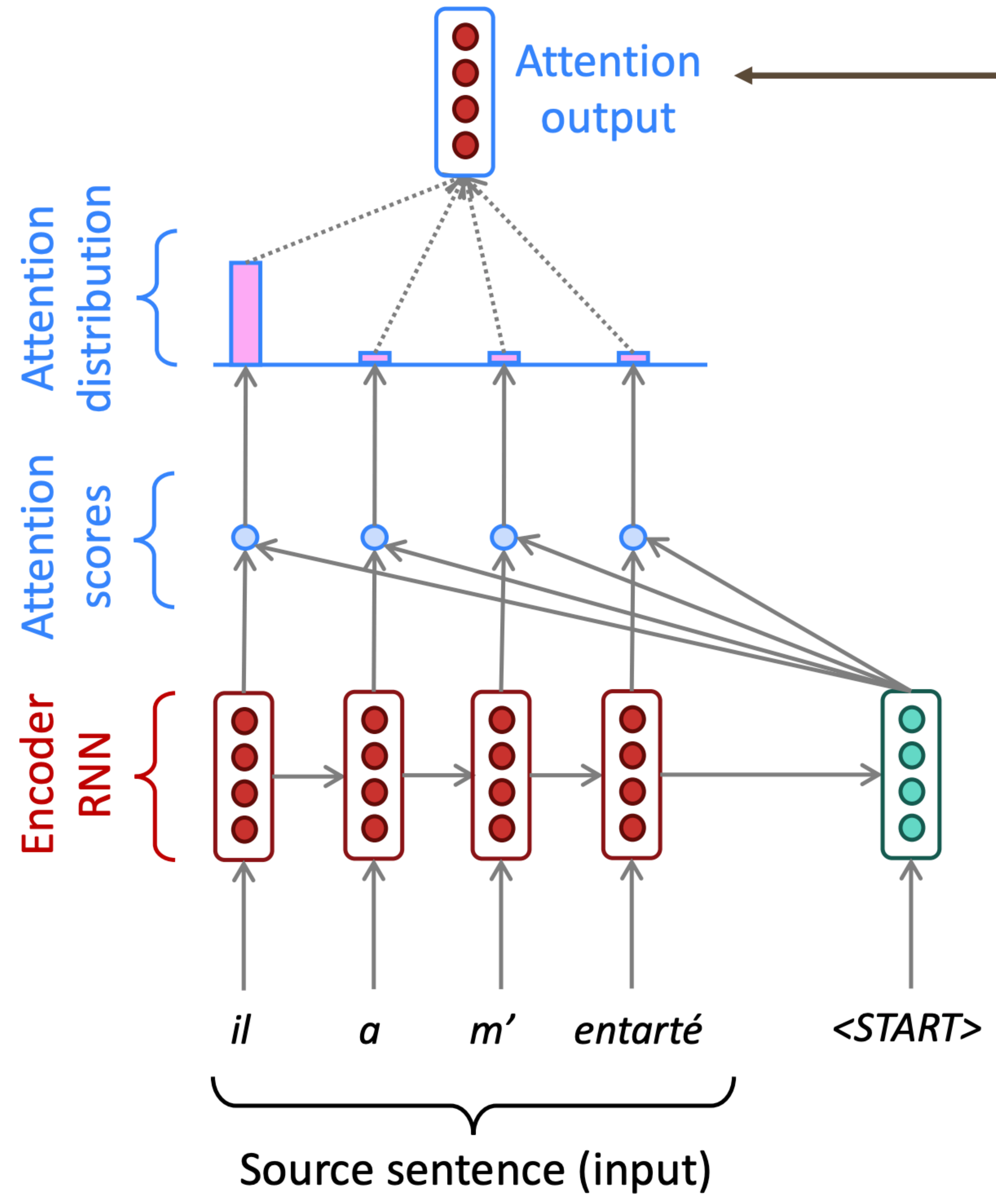
With RNN's



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
il
a
m'
entarté
<START>
Source sentence (input)

Simple Walkthrough Attention

With RNN's

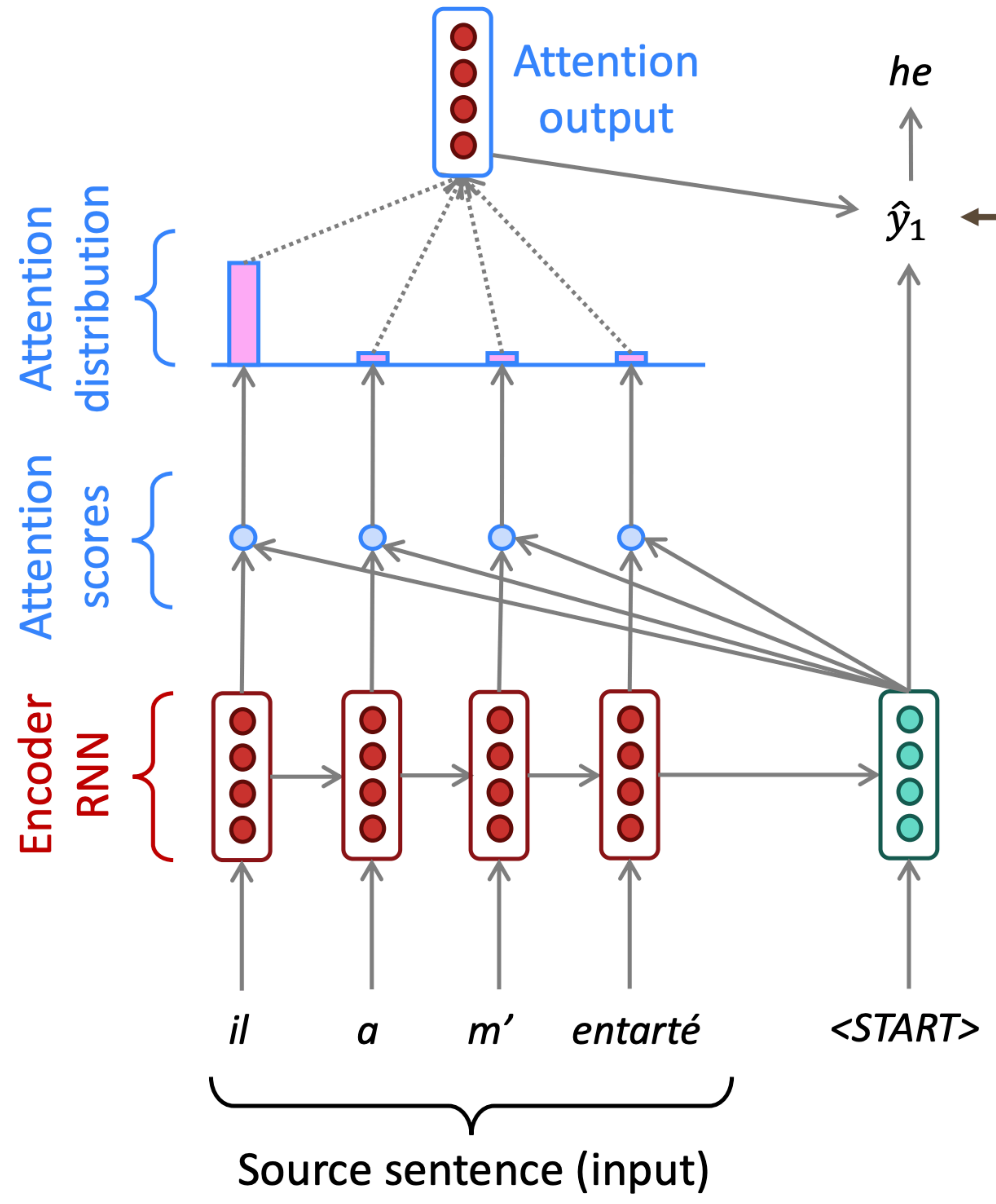


Use the attention distribution to take a **weighted sum** of the encoder hidden states.

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

Simple Walkthrough Attention

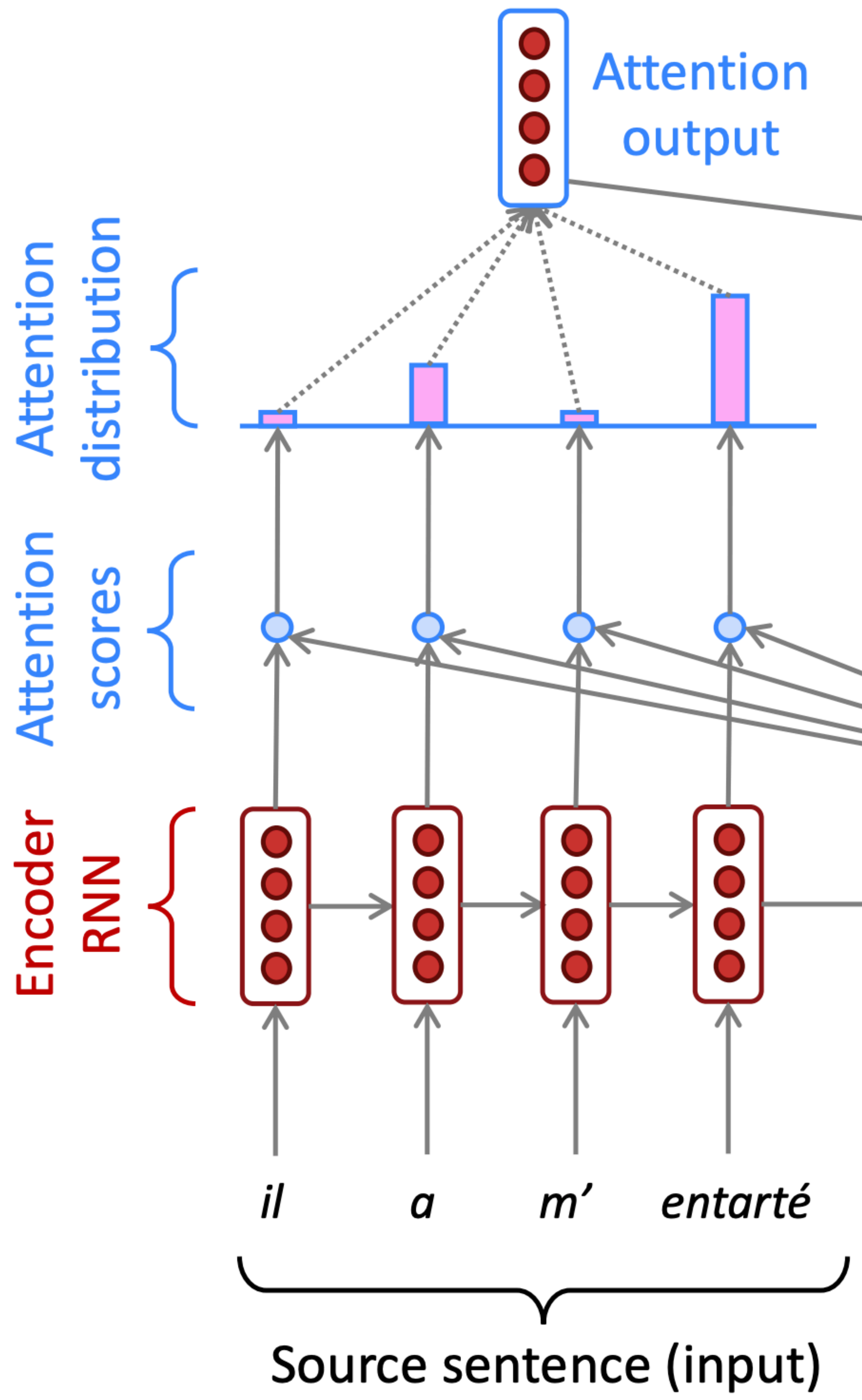
With RNN's



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

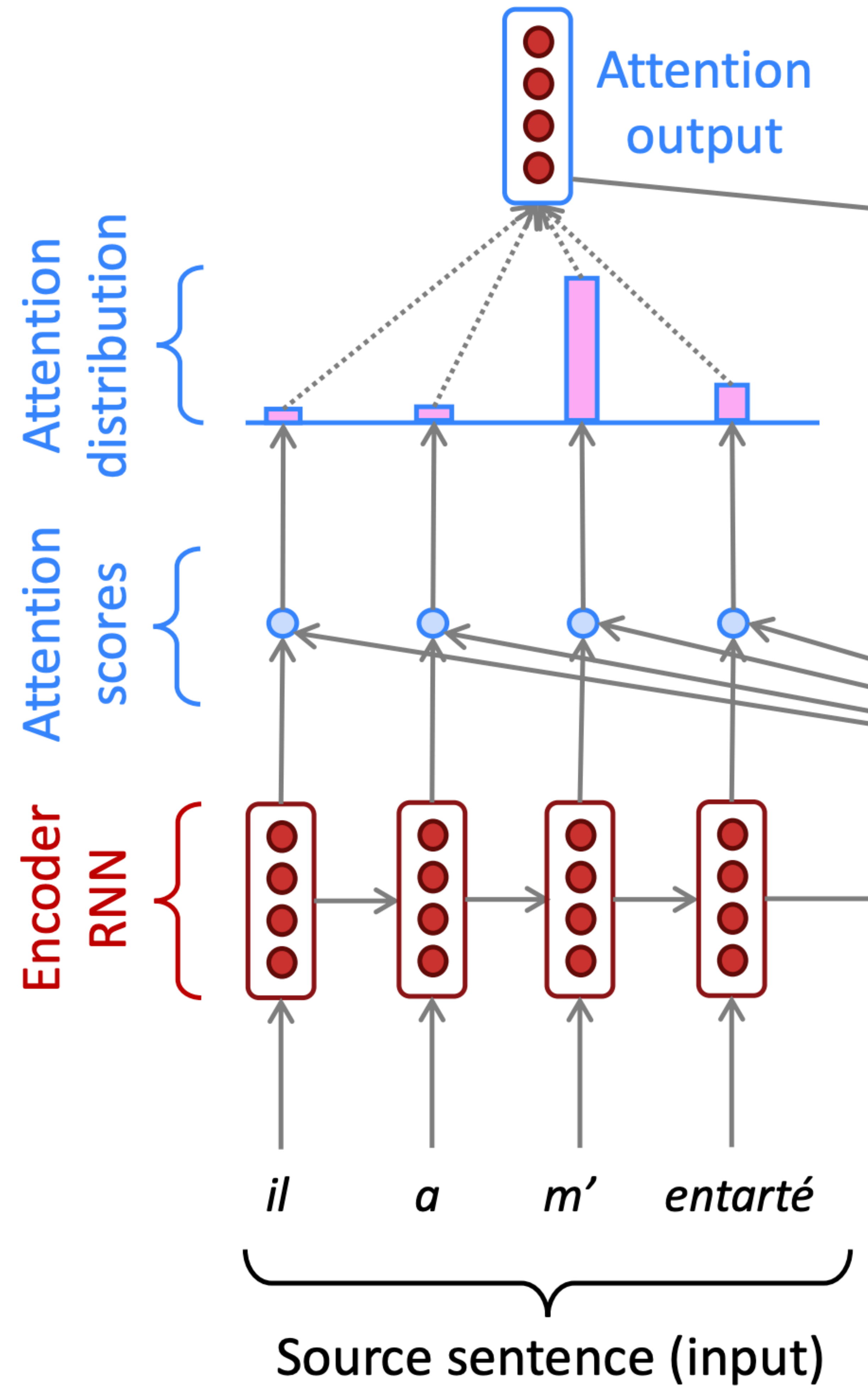
Simple Walkthrough Attention

With RNN's



Simple Walkthrough Attention

With RNN's



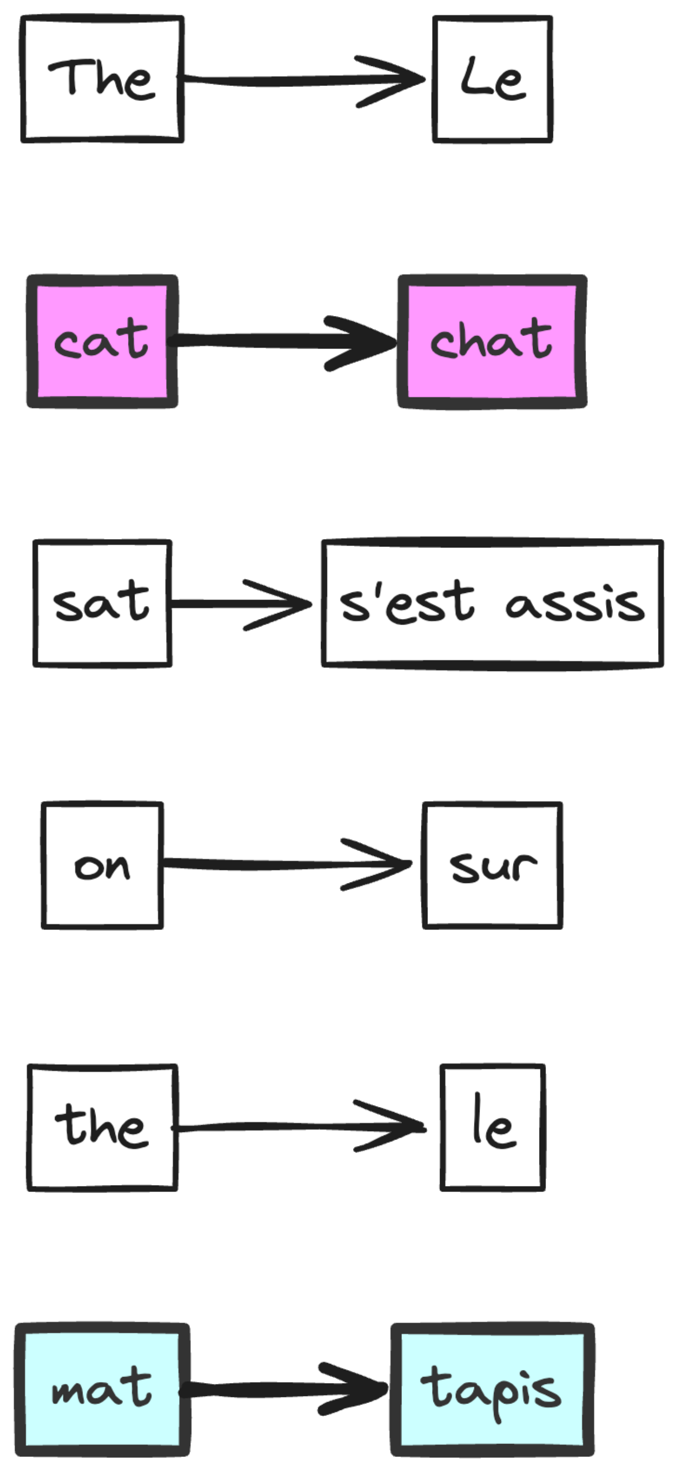
Attention in Machine Translation

An Intuition

Example of translating "The cat sat on the mat" to French

Word-level alignment through attention

Varying attention weights for different words



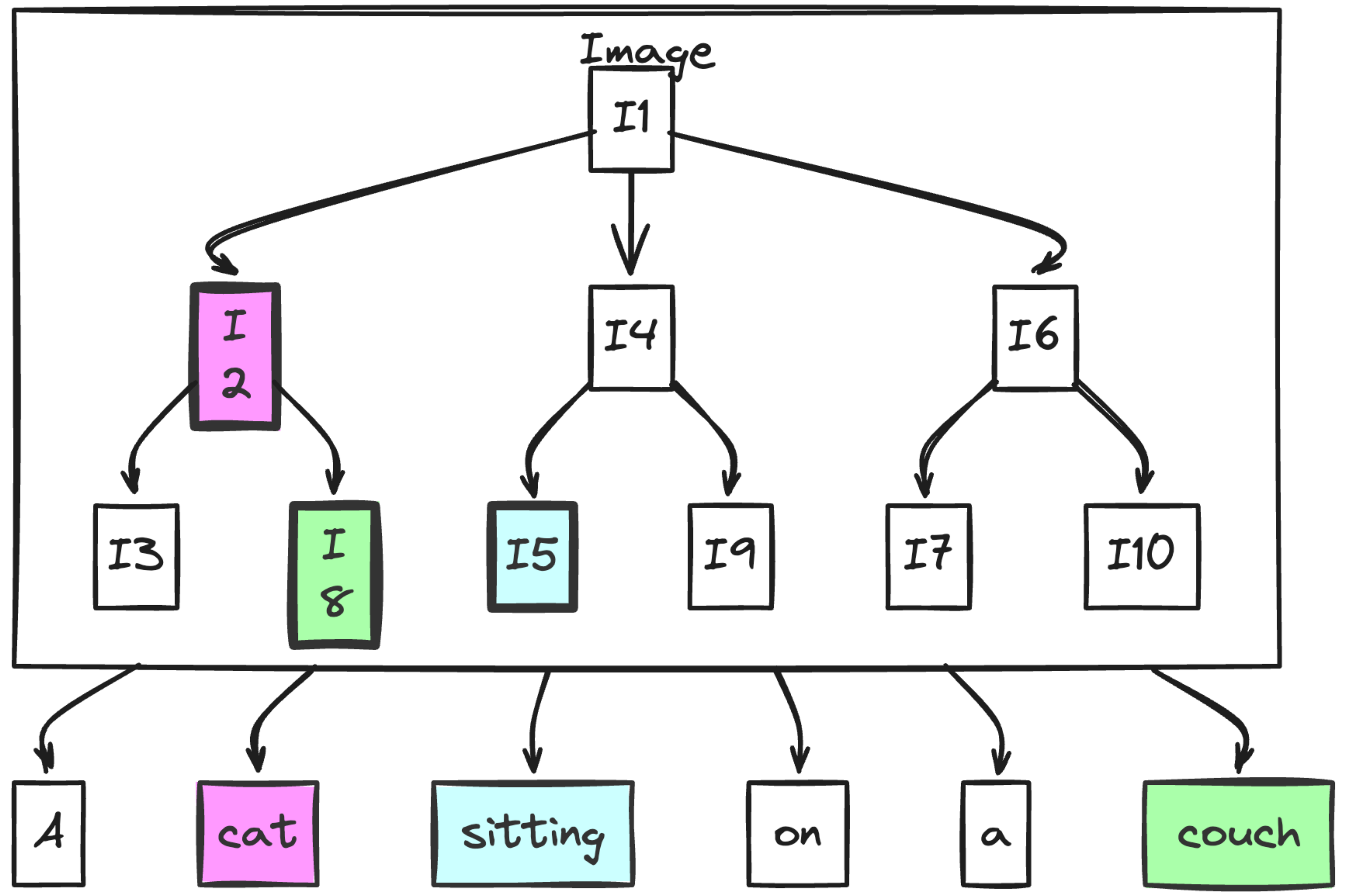
Attention in Vision

An Intuition

Break image into “units”

Word-patch alignment through attention

Attention is a general purpose operation.

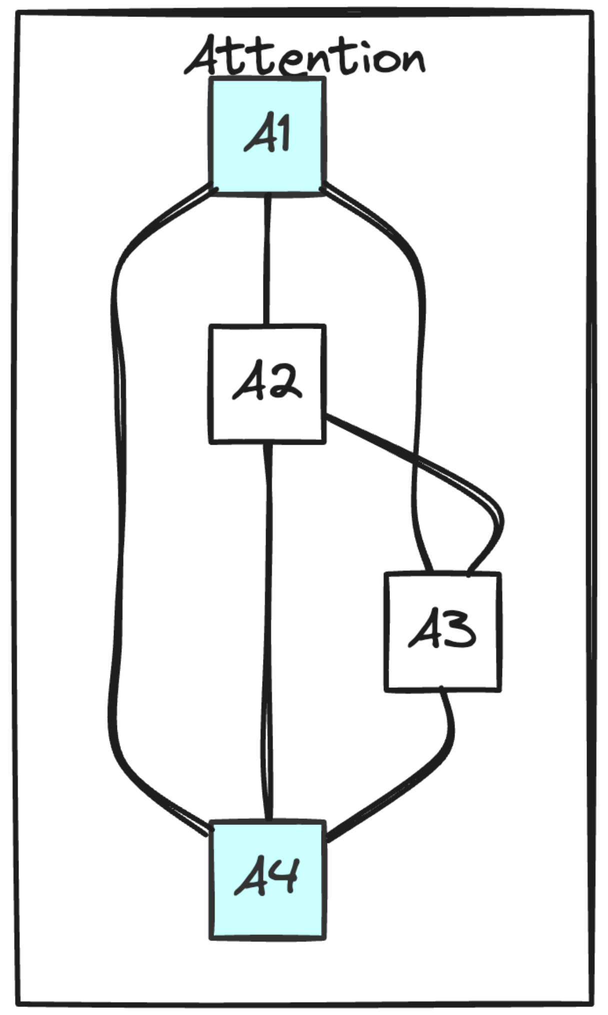


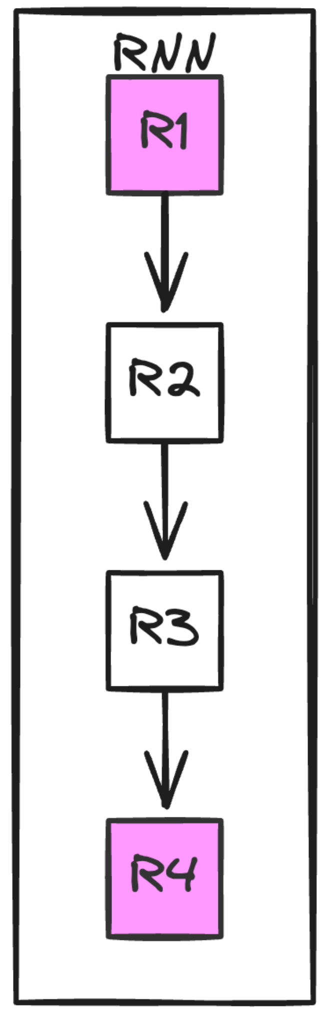
Long-Term Dependencies

RNN vs. Attention

Attention: Direct connections between any two positions in a sequence.

Comparison with RNNs (growing path length)





Formalizing a Simple Attention

Pre-Transformer Era

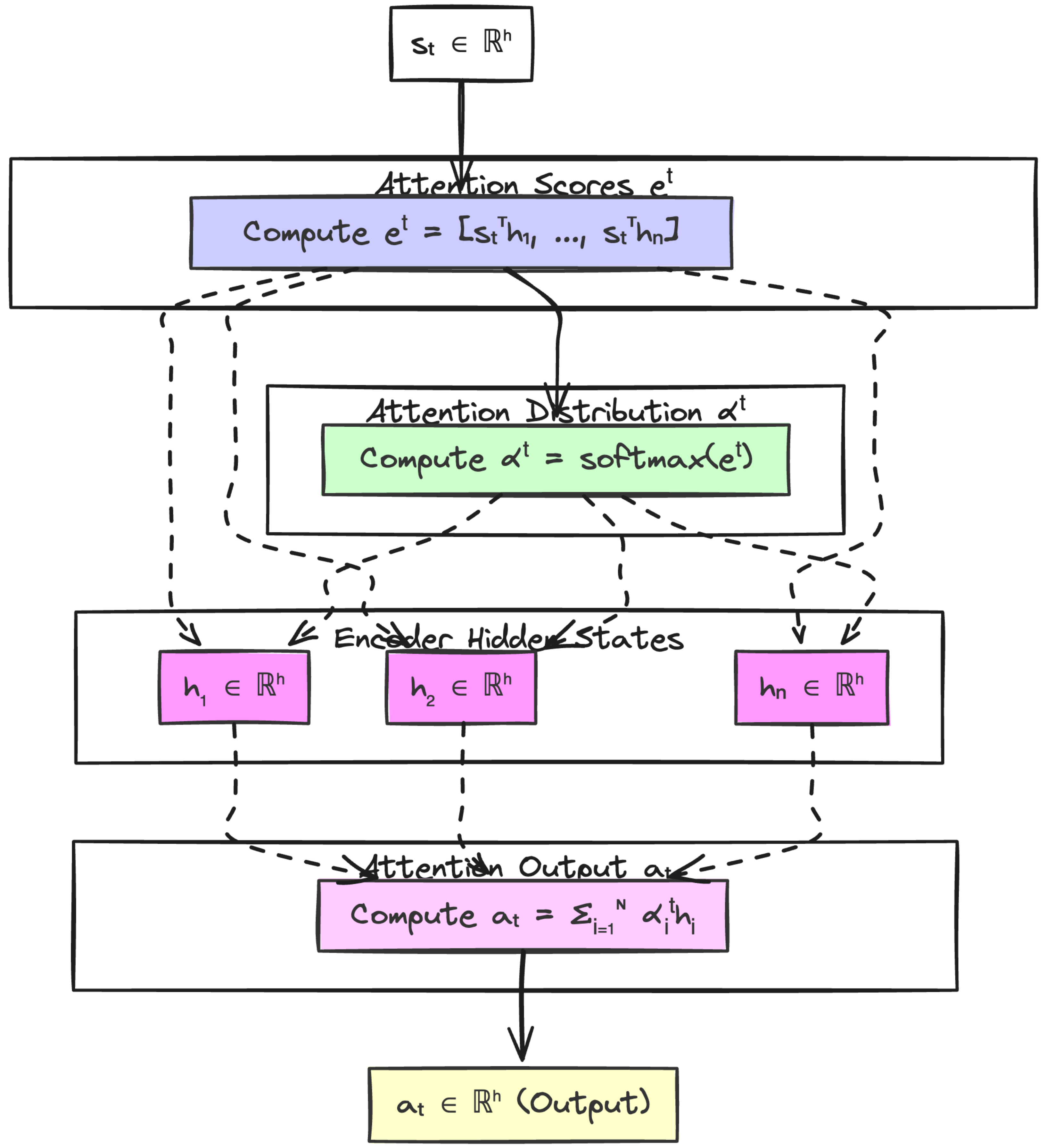
1. We have some hidden states: $h_1, \ldots, h_N \in \mathbb{R}^h$

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

3. We get the attention scores for this step: $e^t = [s_t^T h_1, \ldots, s_t^T h_N] \in \mathbb{R}^N$

4. We take softmax to get the attention distribution for this step (this is a probability distribution and sums to 1): $\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$

5. We use to take a weighted sum of the encoder hidden states to get the attention output: $a_t = \sum_{i=1}^{N} \alpha_i^t h_i \in \mathbb{R}^h$



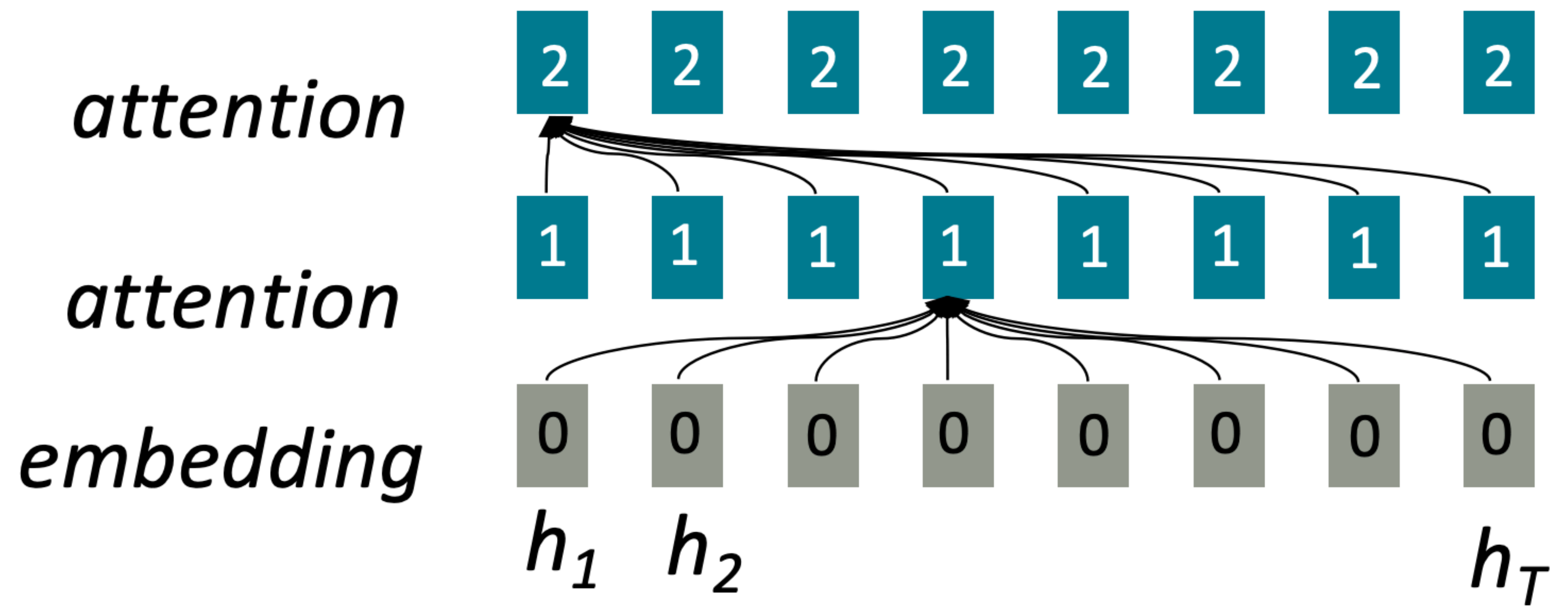
$s_t \in \mathbb{R}^h$
Attention Scores e^t
Compute $e^t = [s_t^T h_1, \ldots, s_t^T h_n]$
Attention Distribution α^t
Compute α^t = softmax(e^t)
Encoder Hidden States
$h_1 \in \mathbb{R}^h$
$h_2 \in \mathbb{R}^h$
$h_n \in \mathbb{R}^h$
Attention Output a_t
Compute $a_t = \Sigma_{i=1}^N \alpha_i^t h_i$
$a_t \in \mathbb{R}^h$ (Output)

Attention Is Parallelizable

Attention treats each word's representation as a query to access and incorporate information from a set of values.

Number of unparallelizable operations does not increase with sequence length.

Maximum interaction distance: O(1), since all words interact at every layer!



All words attend to all words in previous layer; most arrows here are omitted

Attention Is Great!

Attention significantly **improves sequence modeling performance**

- It's very useful to allow decoder to focus on certain parts of the source

Attention provides **a more "human-like" model** of the MT process

- You can look back at the source sentence while translating, rather than needing to remember it all

Attention **solves the bottleneck problem**

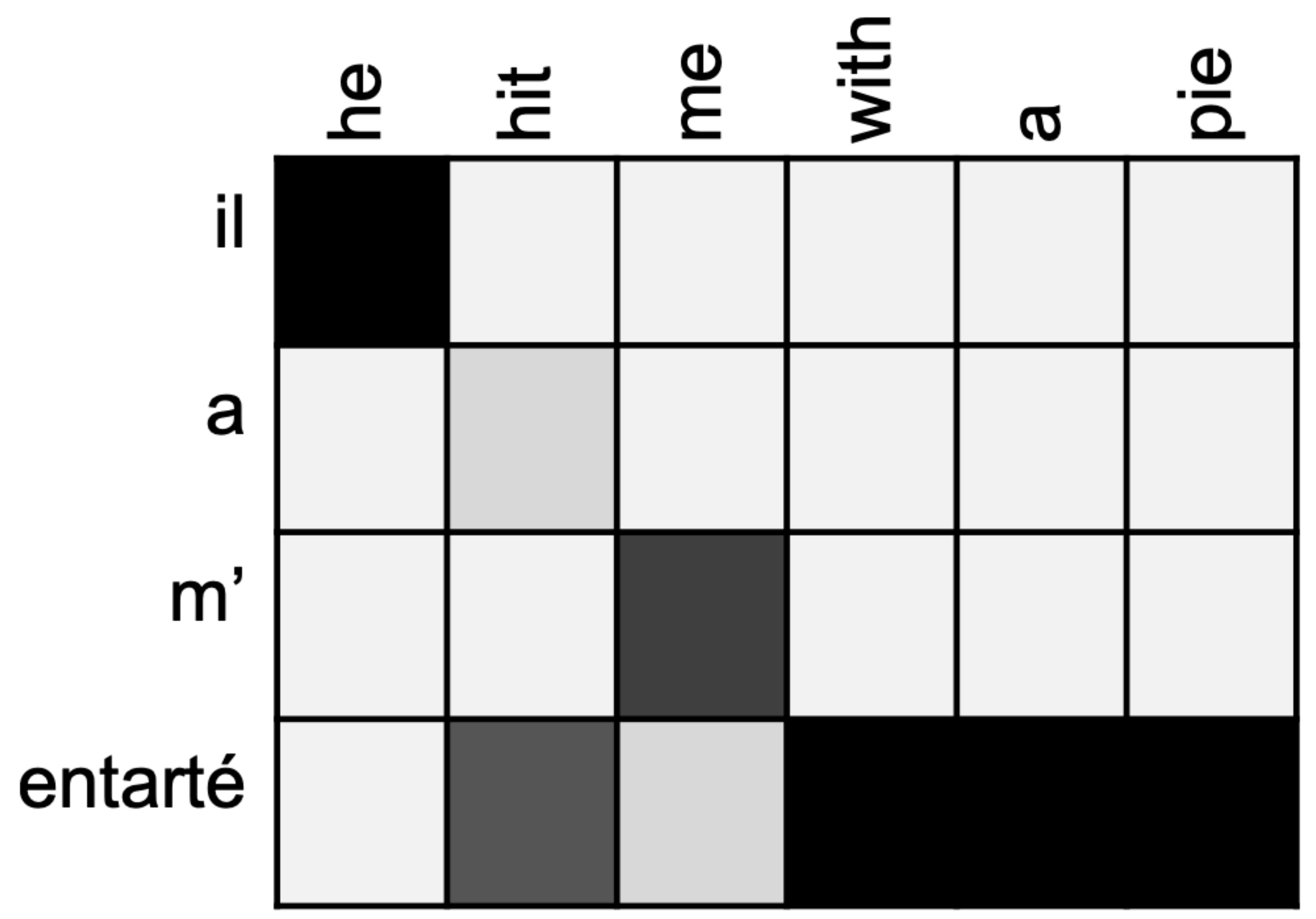
- Attention allows decoder to look directly at source; bypass bottleneck

Attention **helps with the vanishing gradient problem**

- Provides shortcut to faraway states

Attention **provides some interpretability**

- By inspecting attention distribution, we see what the decoder was focusing on
- We get (soft) alignment for free!
- This is cool because we never explicitly trained an alignment system
- The network just learned alignment by itself



Attention is a *general* Deep Learning technique

More general definition of attention: Given a set of vector **values**, and a vector **query**, attention is a technique to compute a weighted sum of the values, dependent on the query.

Intuition:

- The weighted sum is a selective summary of the information contained in the values, where the query determines which values to focus on.
- Attention is a way to obtain a fixed-size representation of an arbitrary set of representations (the values), dependent on some other representation (the query).

Attention has become the powerful, flexible, general way pointer and memory manipulation in all deep learning models. An old idea from roughly 2010! From NMT!

Do we need recurrence?

Abstractly: Attention is a way to pass information from a sequence (x) to a neural network input (h_t).

- This is also exactly what RNNs are used for – to pass information!
- Can we just get rid of the RNN entirely? Maybe attention is just a better way to pass information!