Attention in Seq2Seq Models

Tanmoy Chakraborty
Associate Professor, IIT Delhi
https://tanmoychak.com/

Introduction to Large Language Models

NMT: The First Big Success Story of NLP Deep Learning

Neural Machine Translation went from a fringe research attempt in 2014 to the leading standard method in 2016

- 2014: First seq2seq paper published [Sutskever et al. 2014]
- 2016: Google Translate switches from SMT to NMT – and by 2018 everyone had
    - https://www.nytimes.com/2016/12/14/magazine/the-great-ai-awakening.html

- This was amazing!
    - SMT systems, built by hundreds of engineers over many years, were outperformed by NMT systems trained by small groups of engineers in a few months

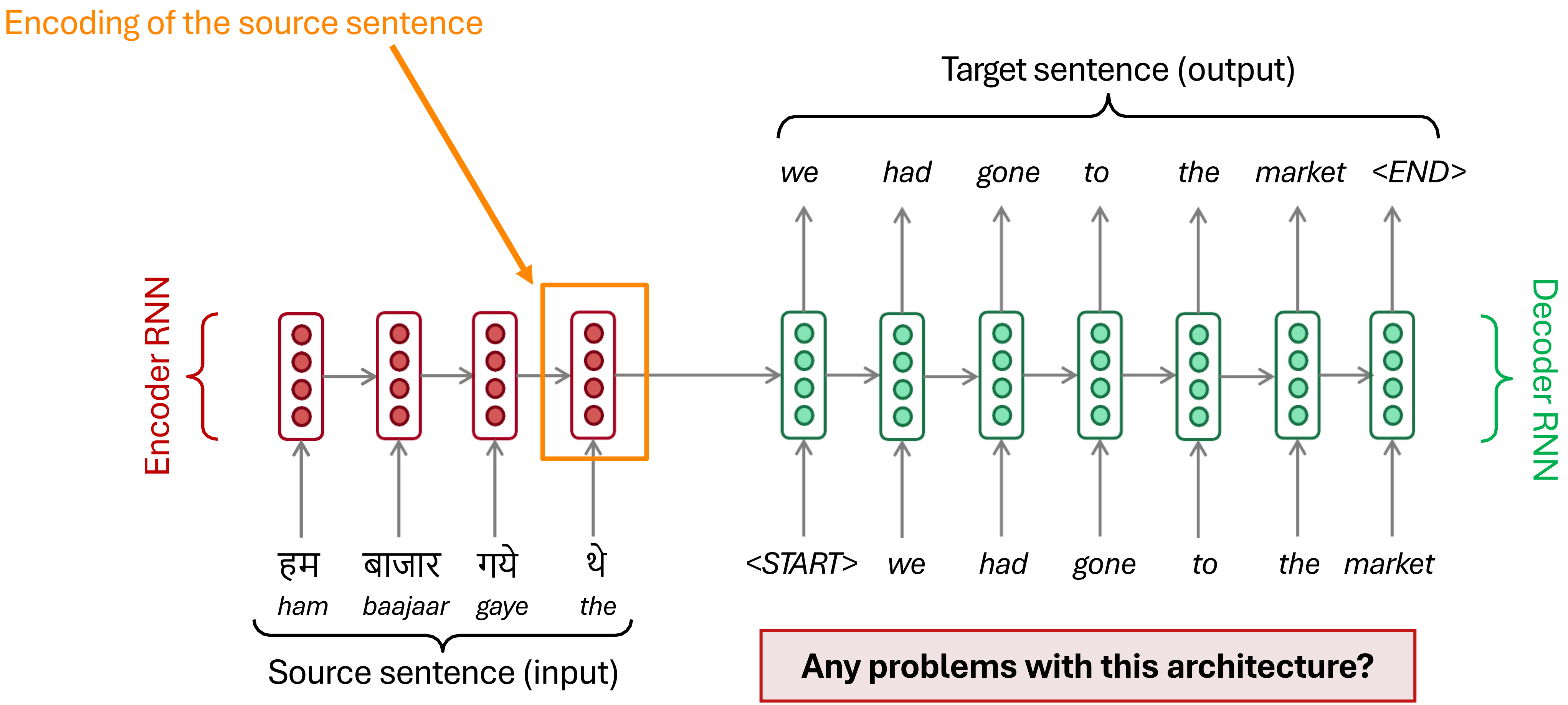
Issues With RNN

- Linear interaction distance
- Bottleneck problem
- Lack of parallelizability

ATTENTION

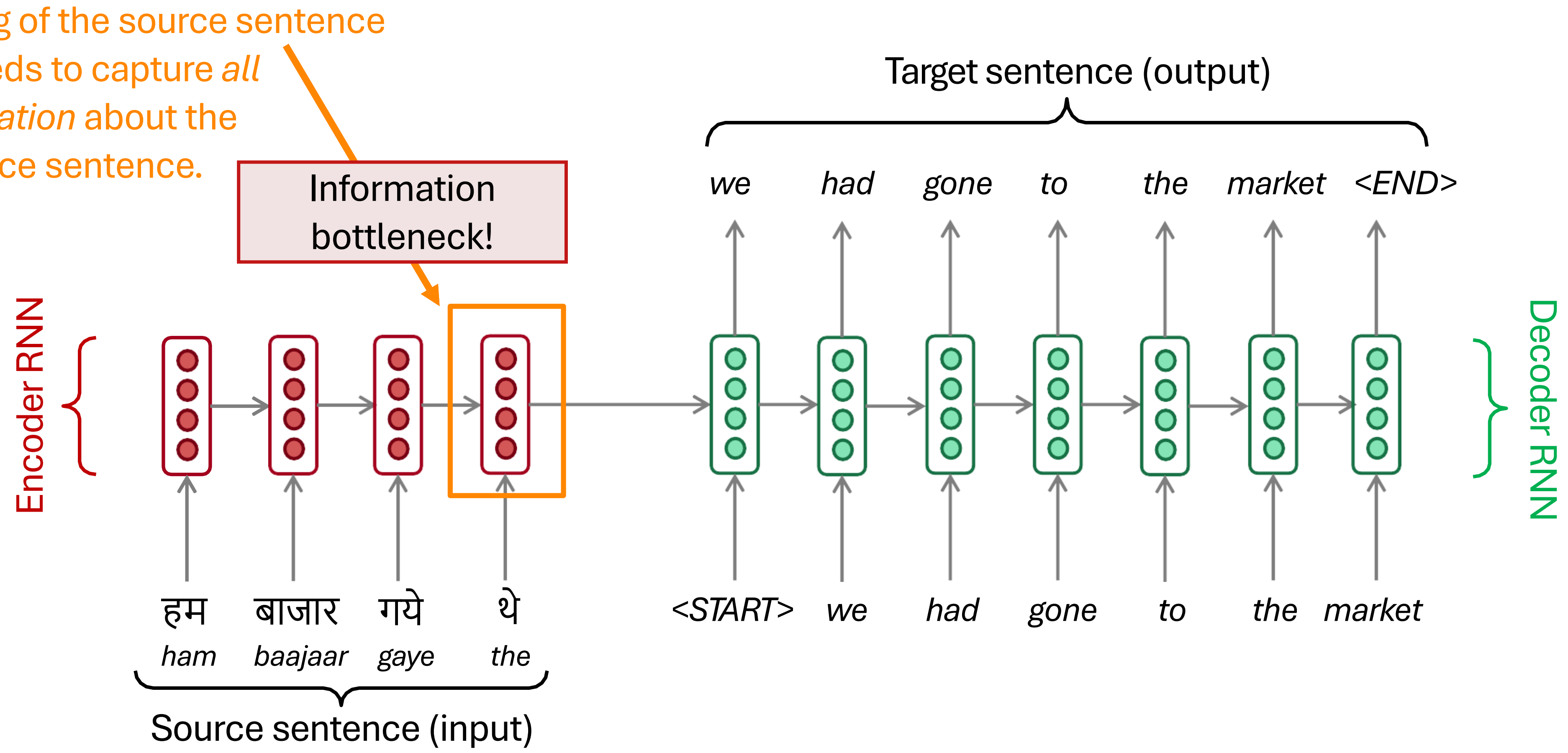
Attention

Sequence-to-Sequence: The Bottleneck Problem



Any problems with this architecture?

Sequence-to-Sequence: The Bottleneck Problem



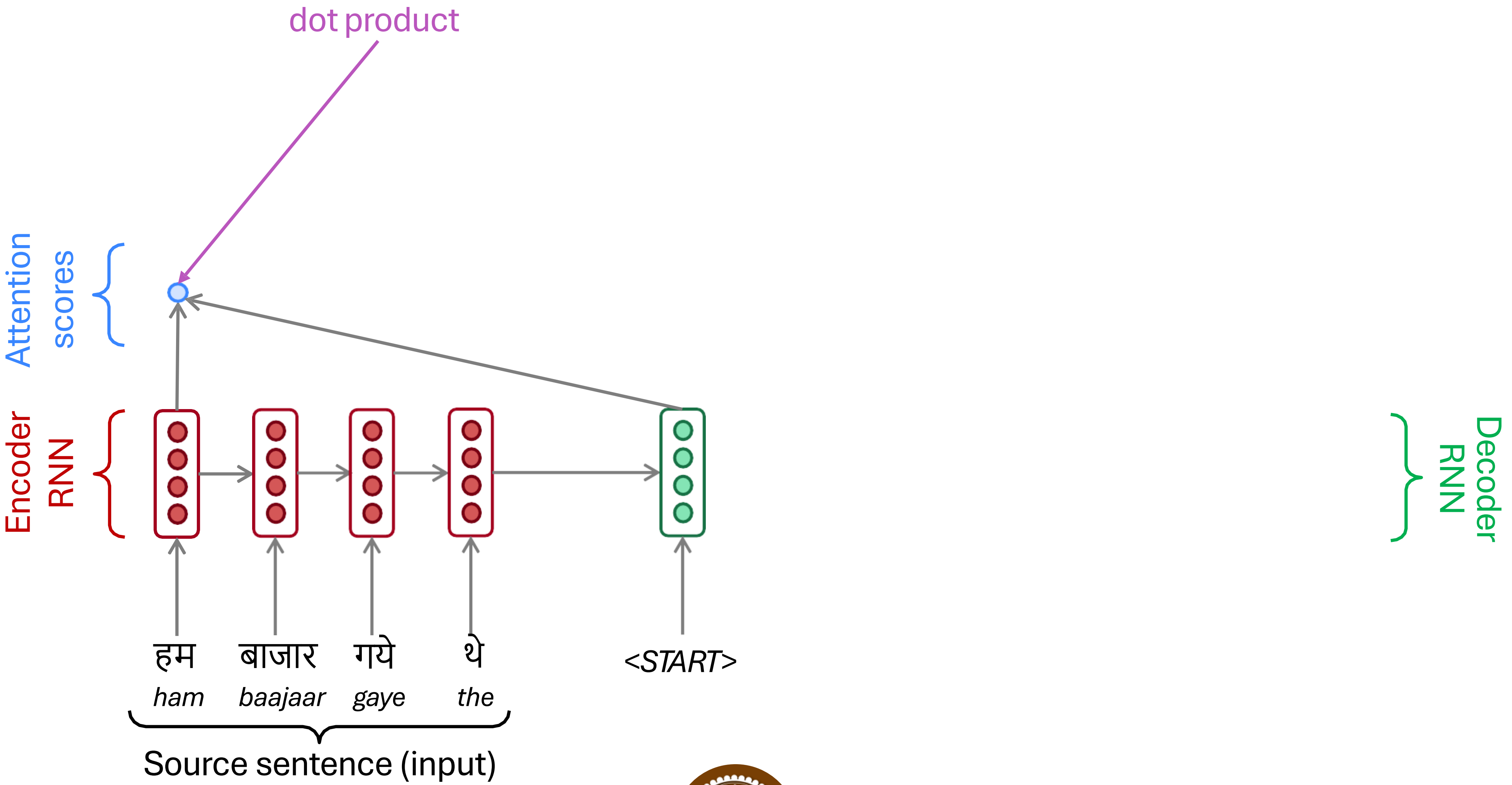
Attention

- **Attention** provides a solution to the bottleneck problem.

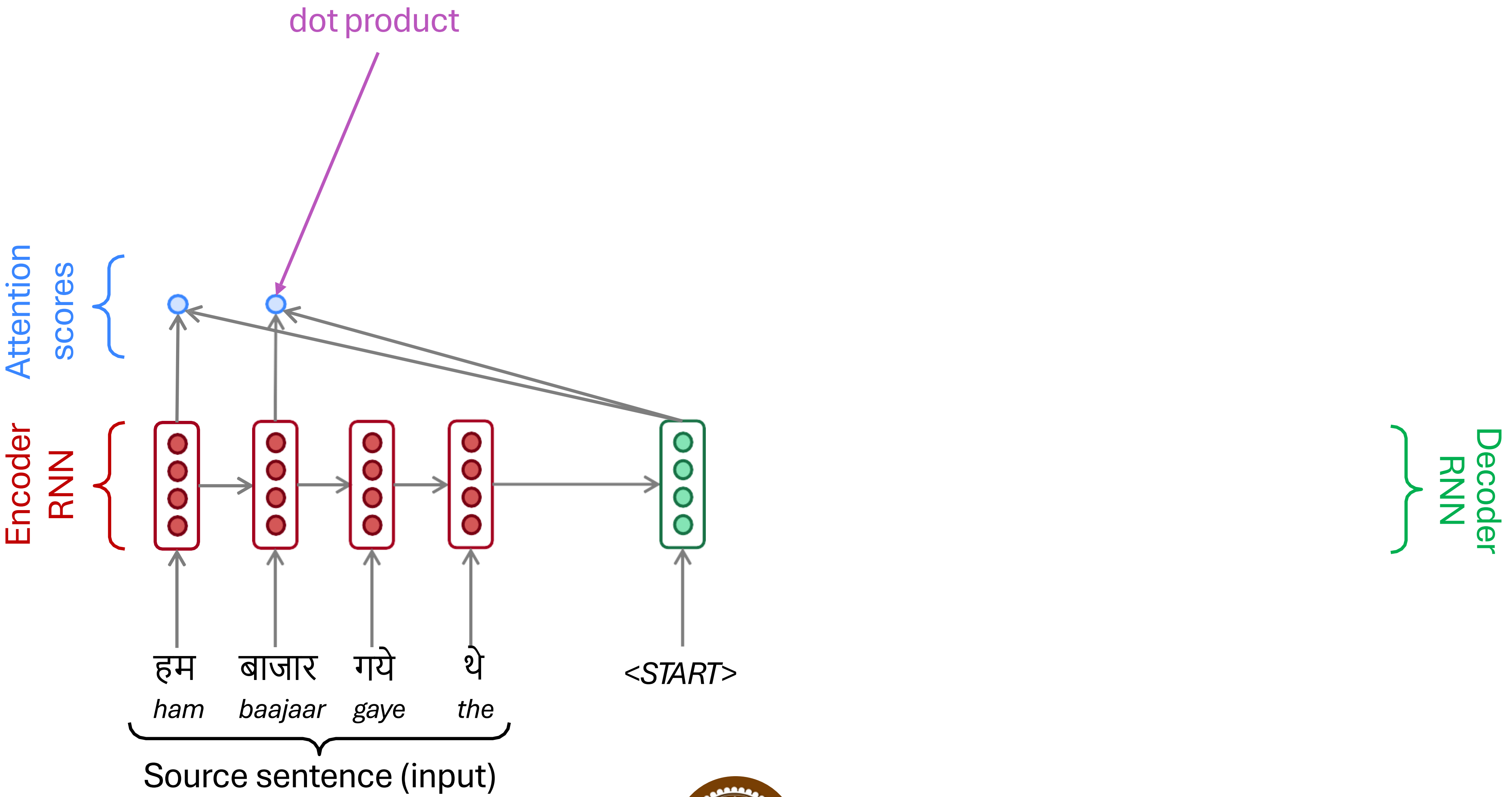
- **Core idea**: on each step of the decoder, use direct connection to the encoder to focus on a particular part of the source sequence

- Let’s start with the visualization of the attention mechanism.

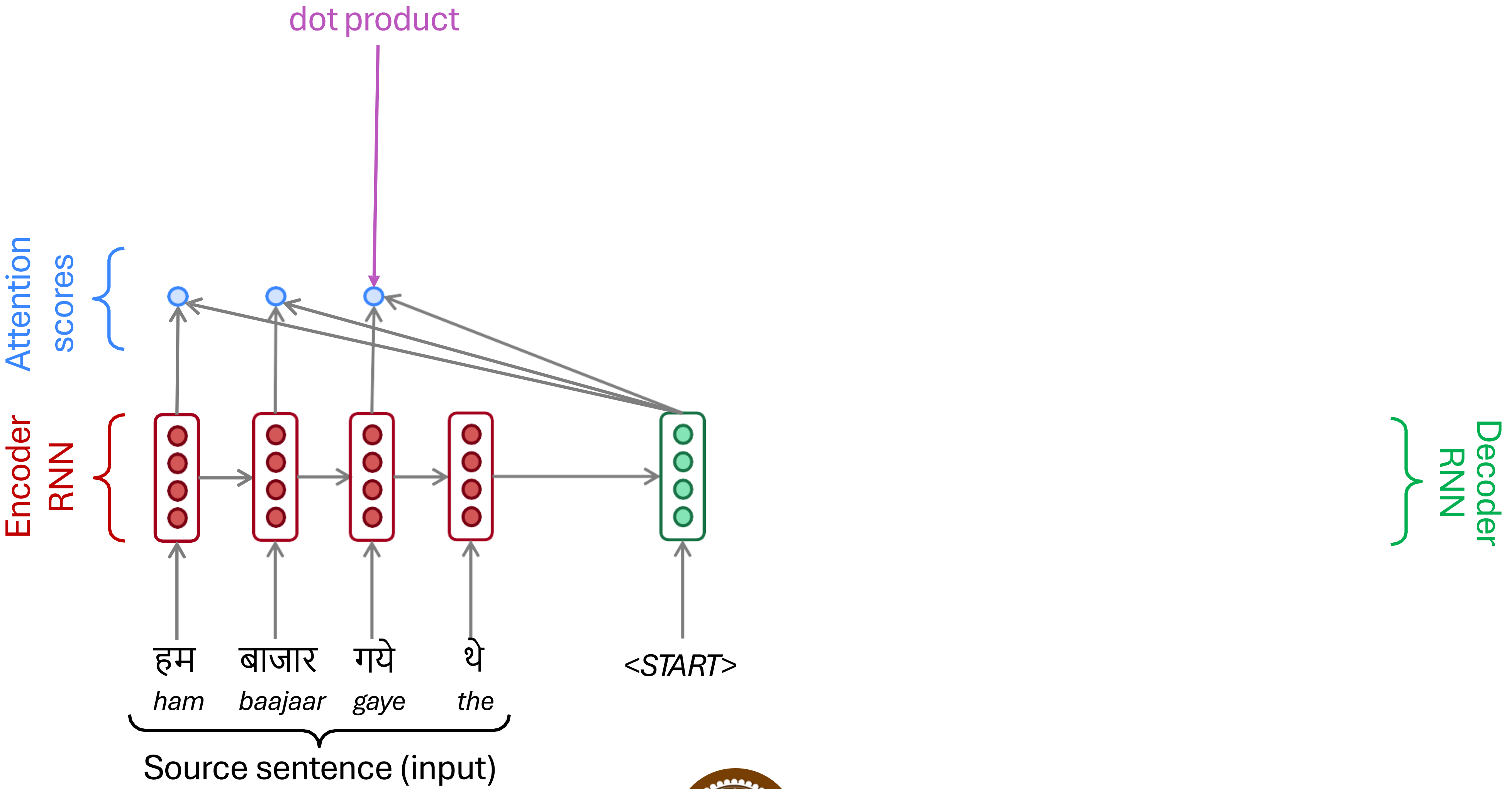
Sequence-to-Sequence With Attention



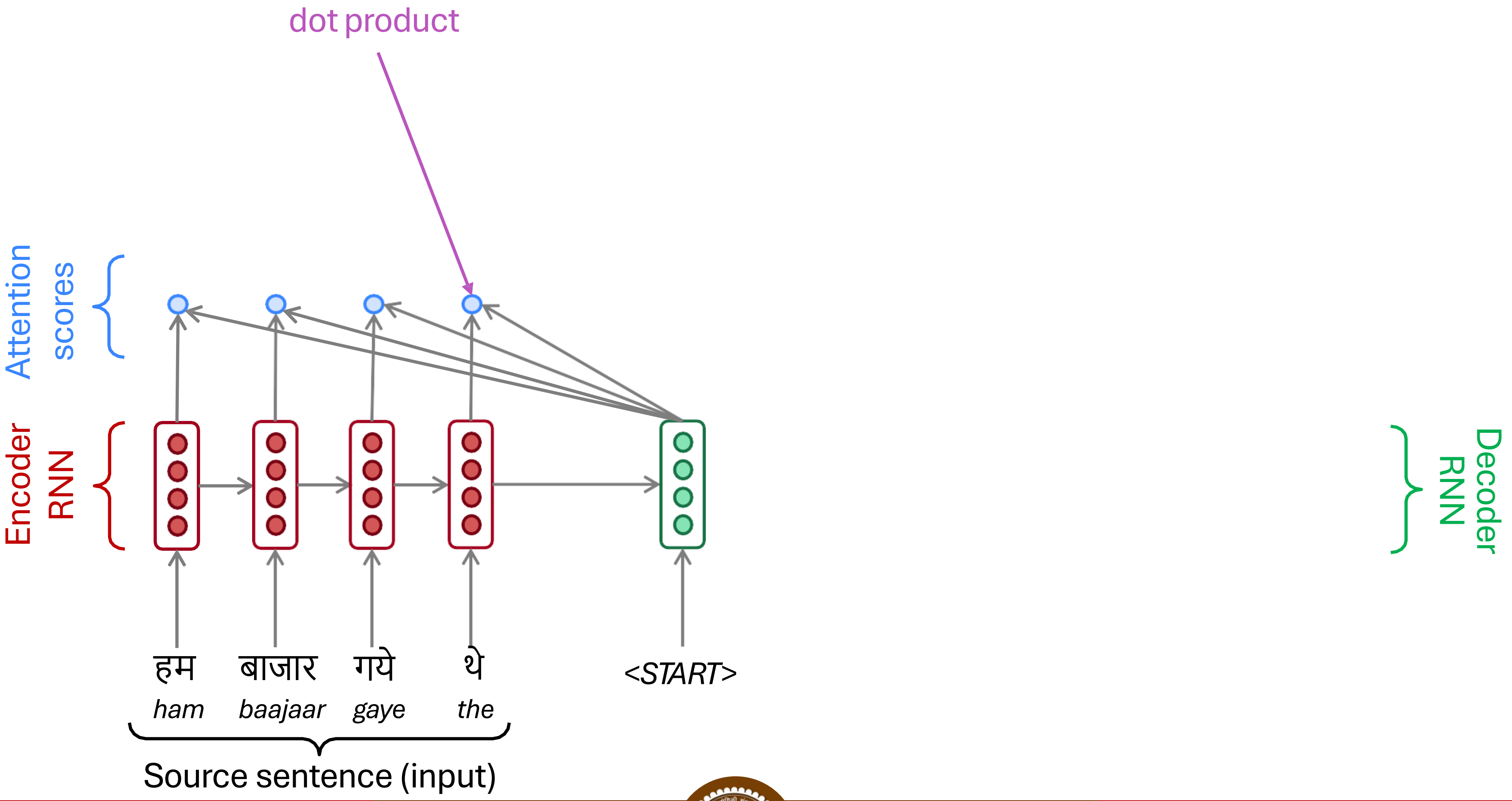
Sequence-to-Sequence With Attention



Sequence-to-Sequence With Attention



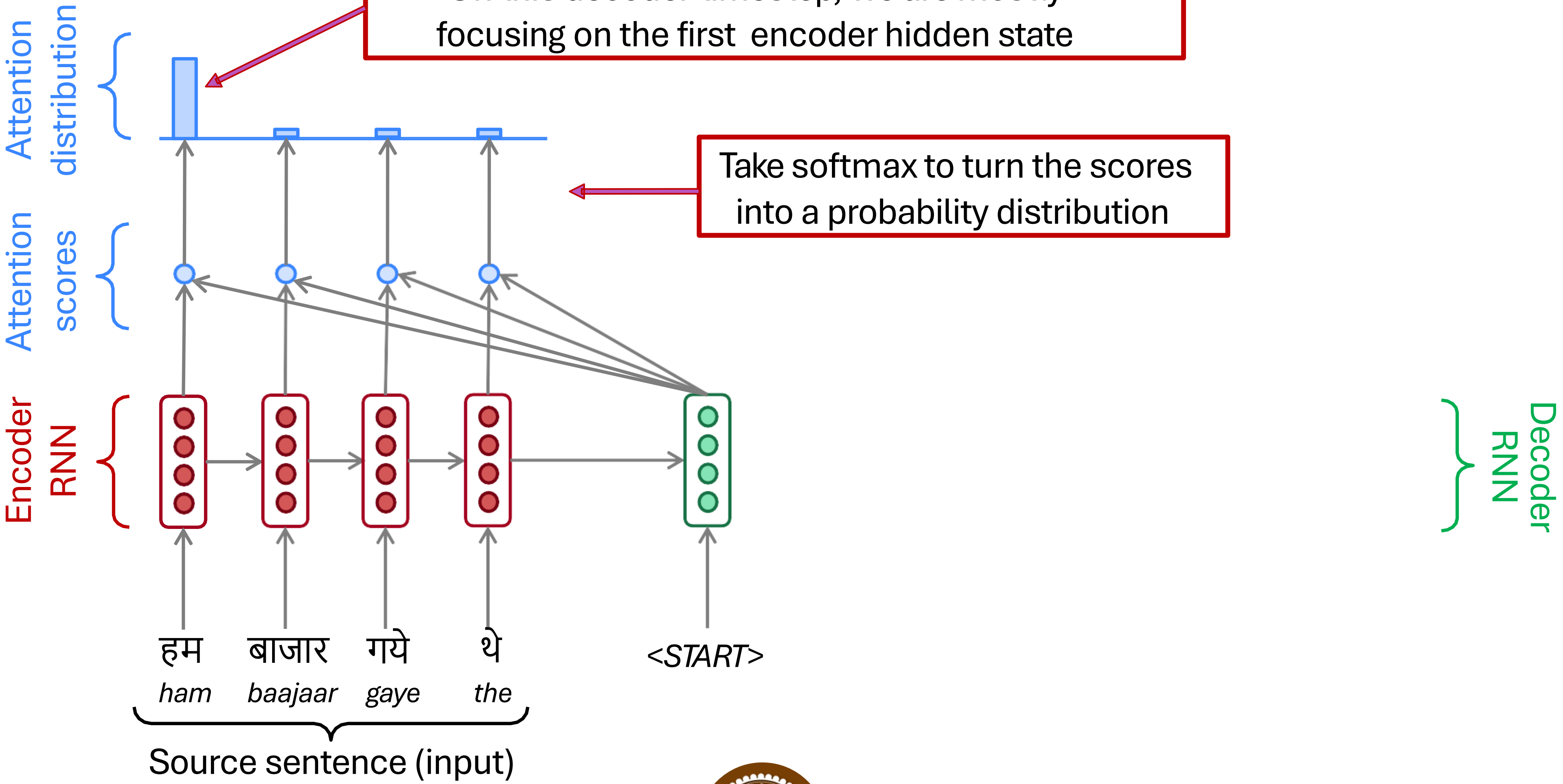
Sequence-to-Sequence With Attention



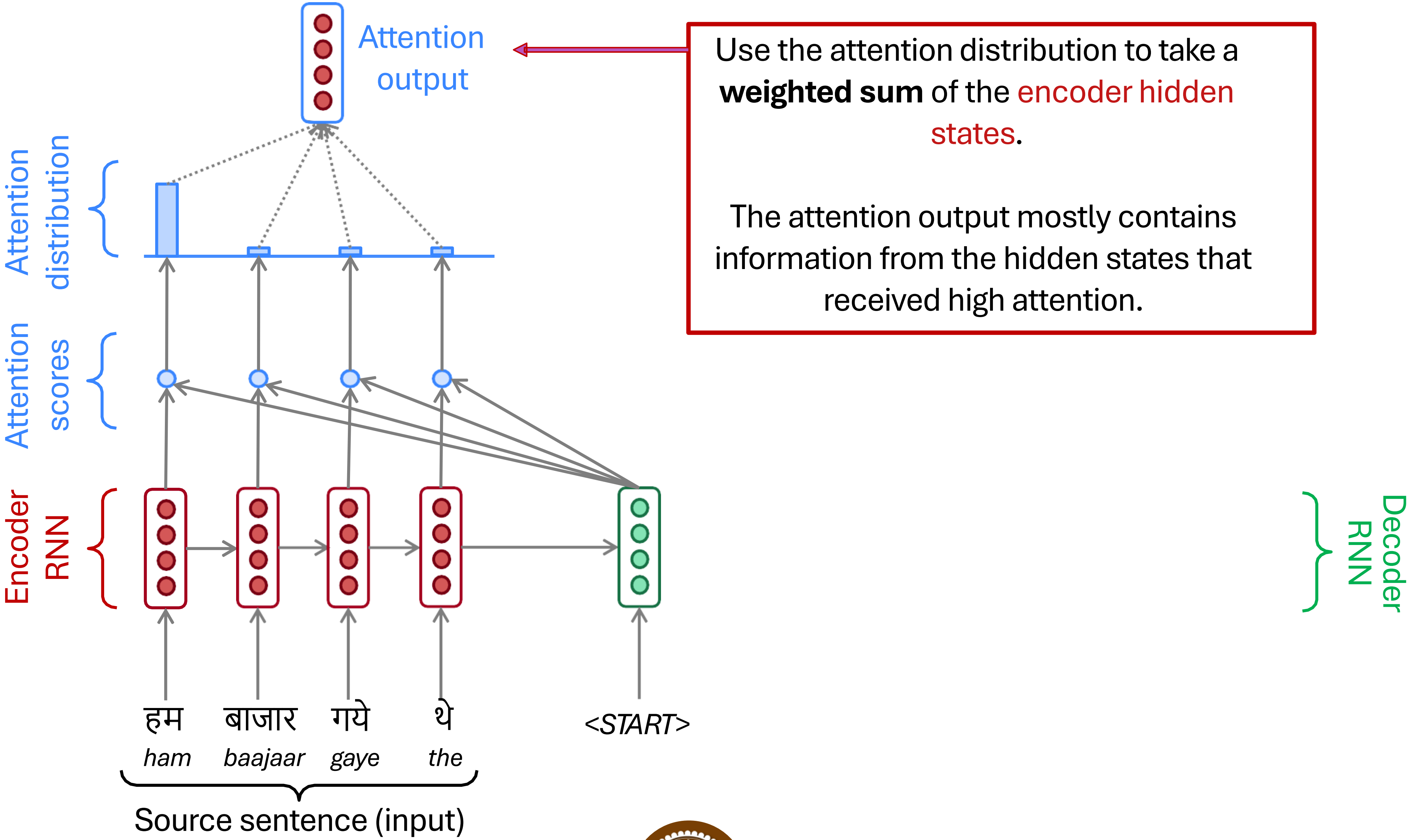
Sequence-to-Sequence With Attention

On this decoder timestep, we are mostly focusing on the first encoder hidden state

Take softmax to turn the scores into a probability distribution



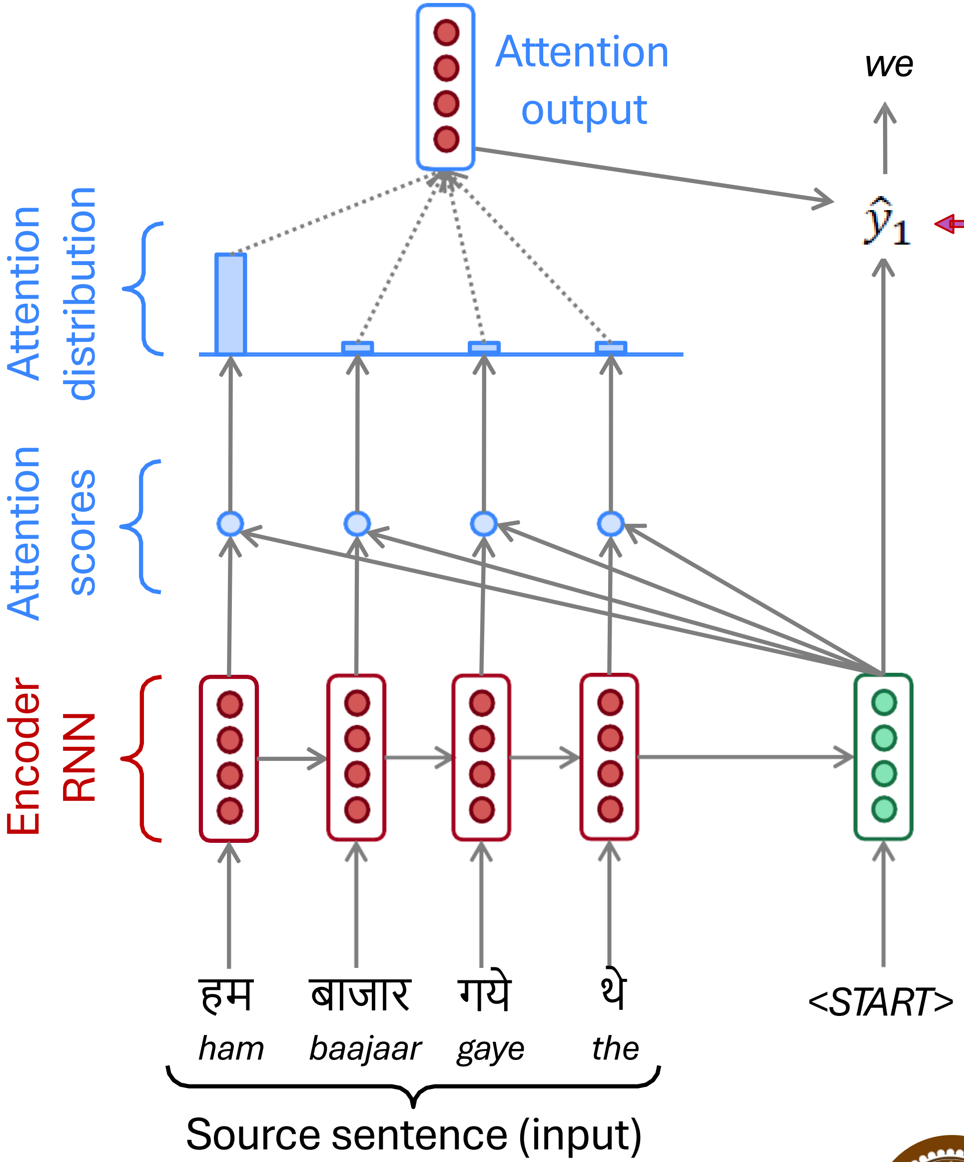
Sequence-to-Sequence With Attention



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

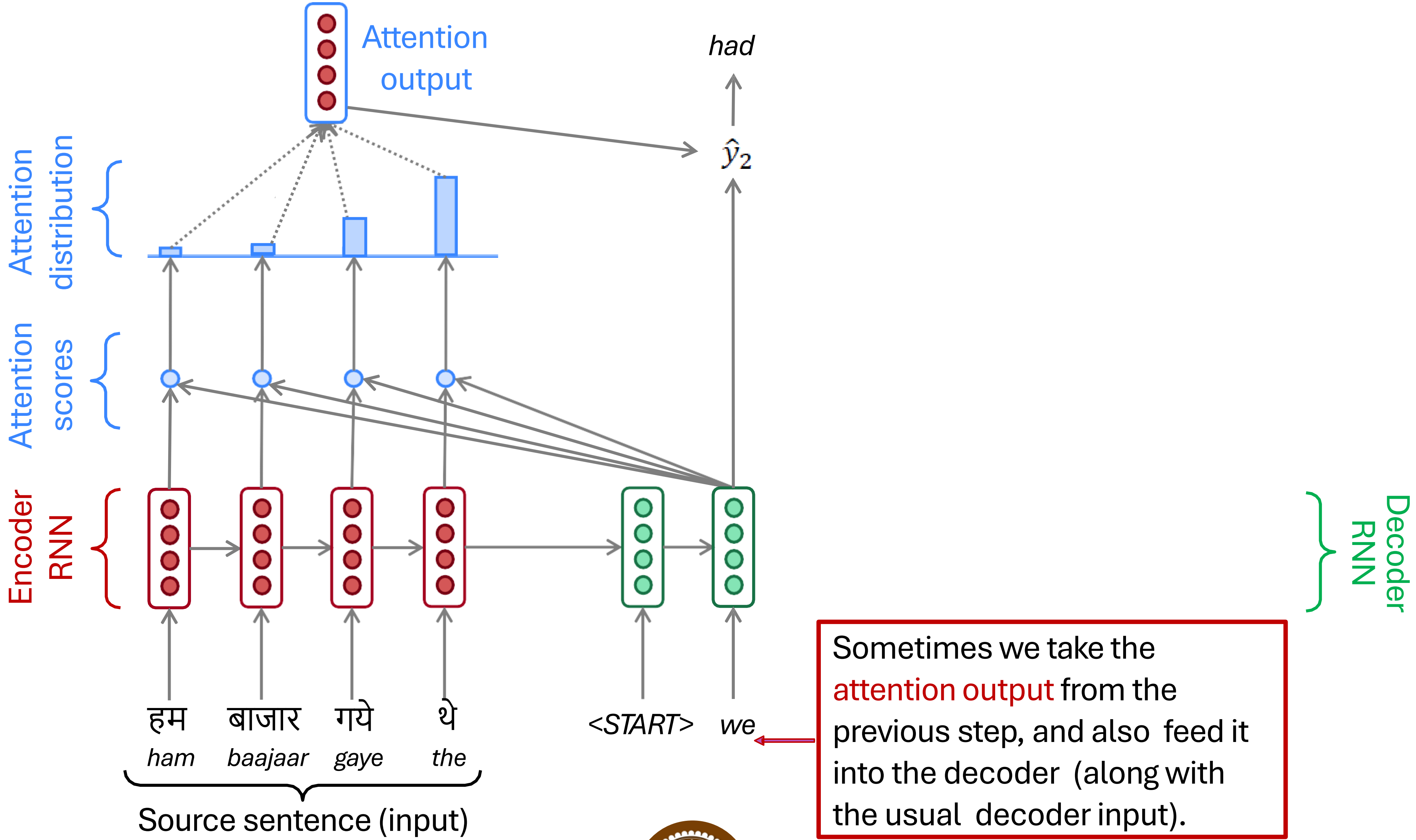
The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-Sequence With Attention



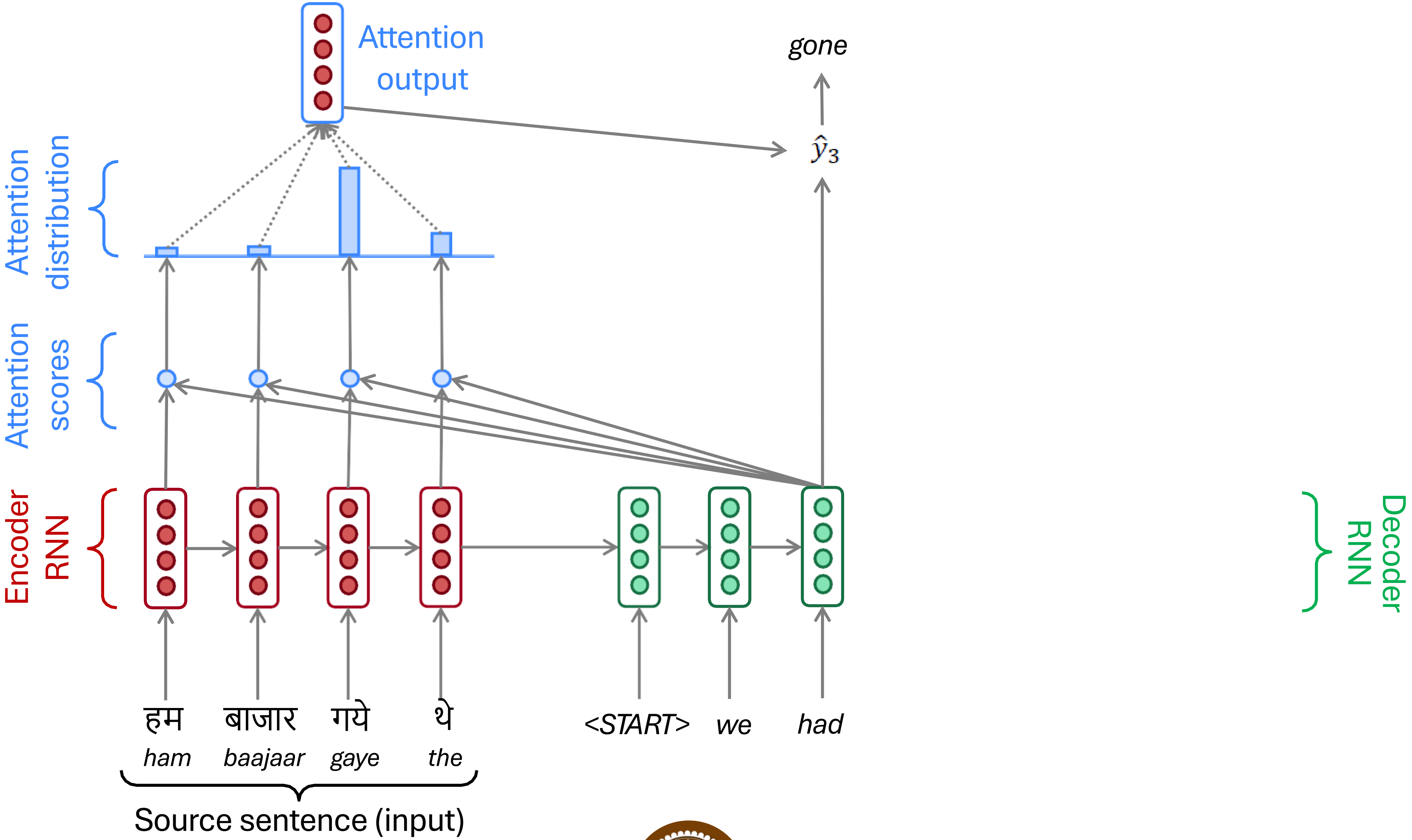
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Sequence-to-Sequence With Attention

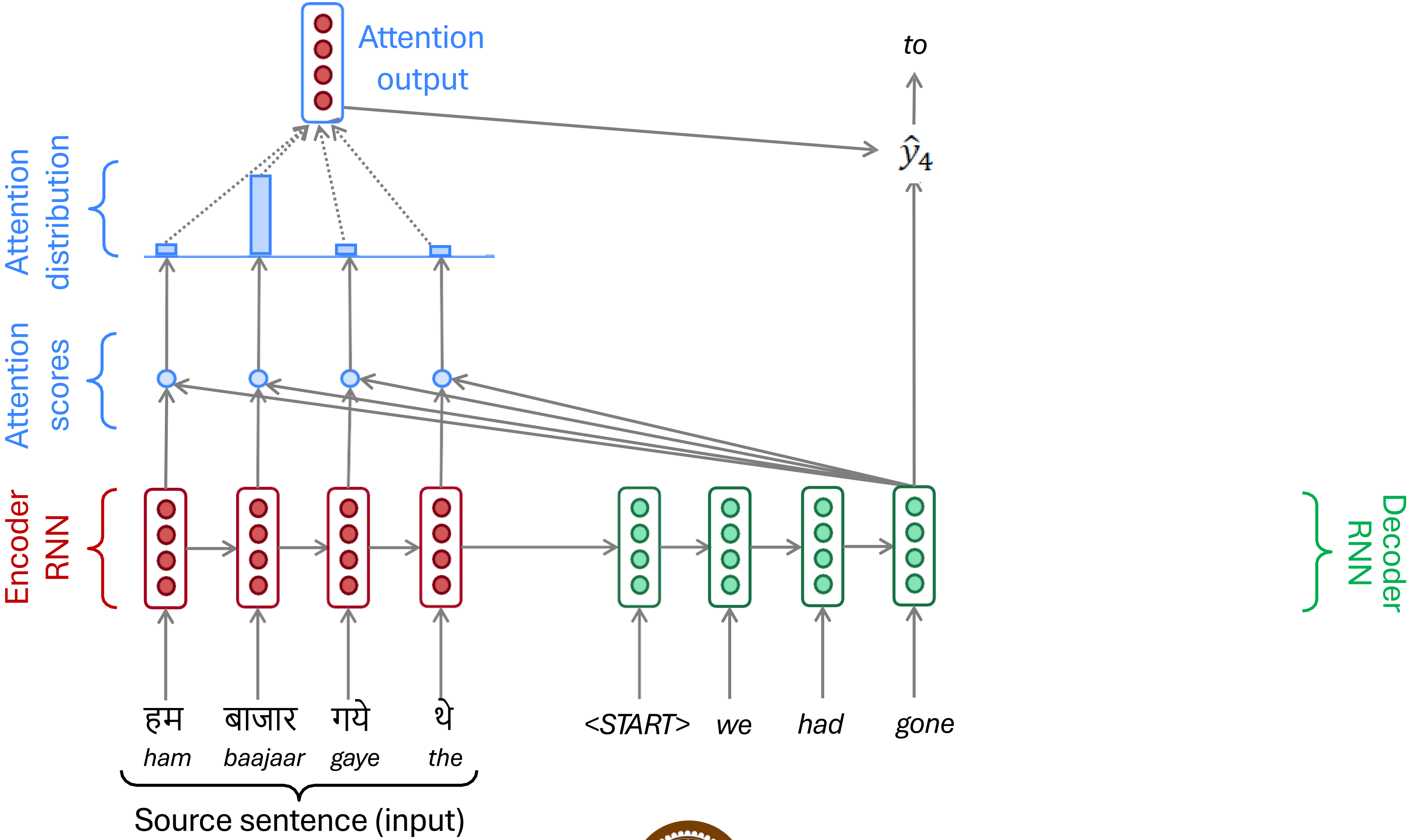


Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input).

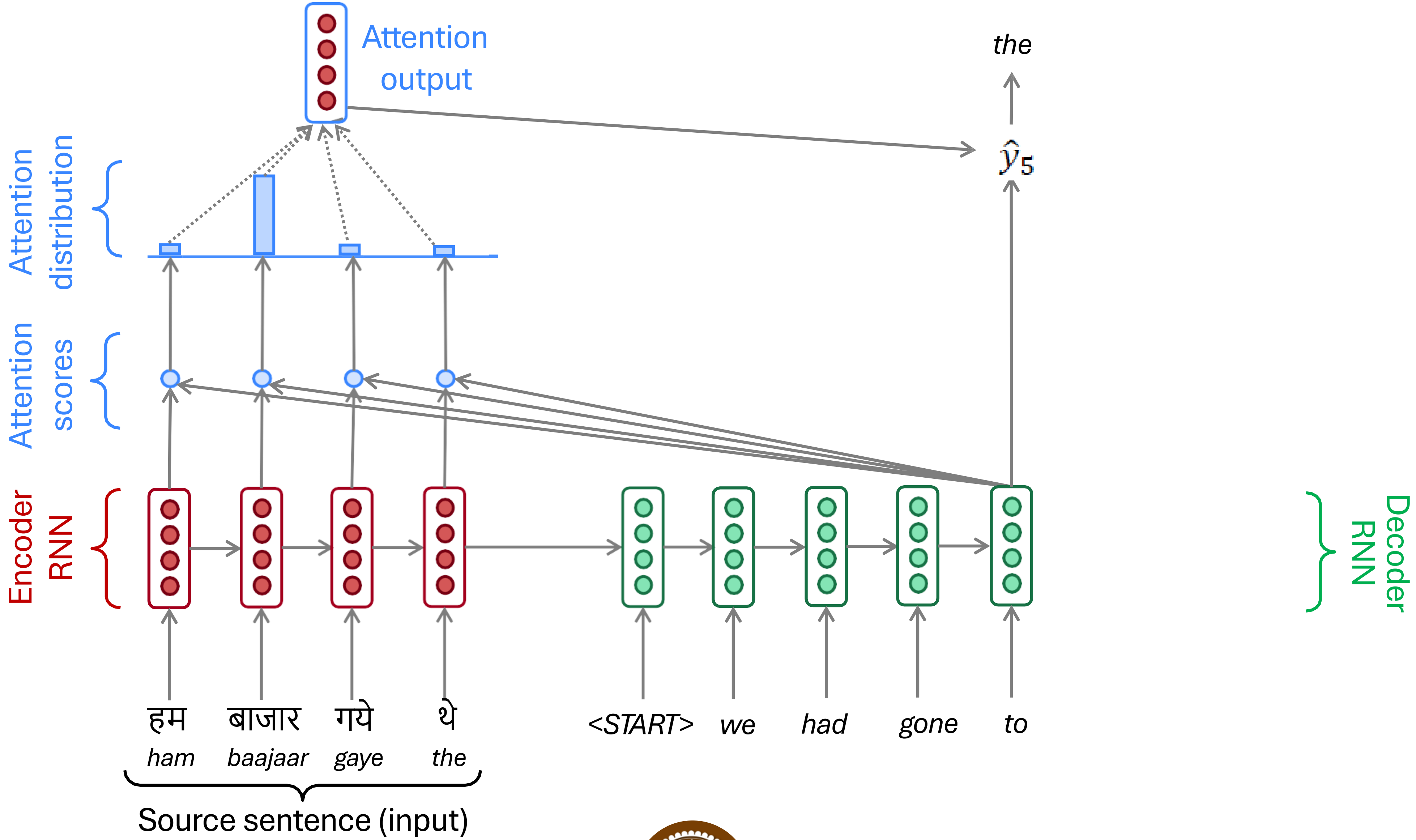
Sequence-to-Sequence With Attention



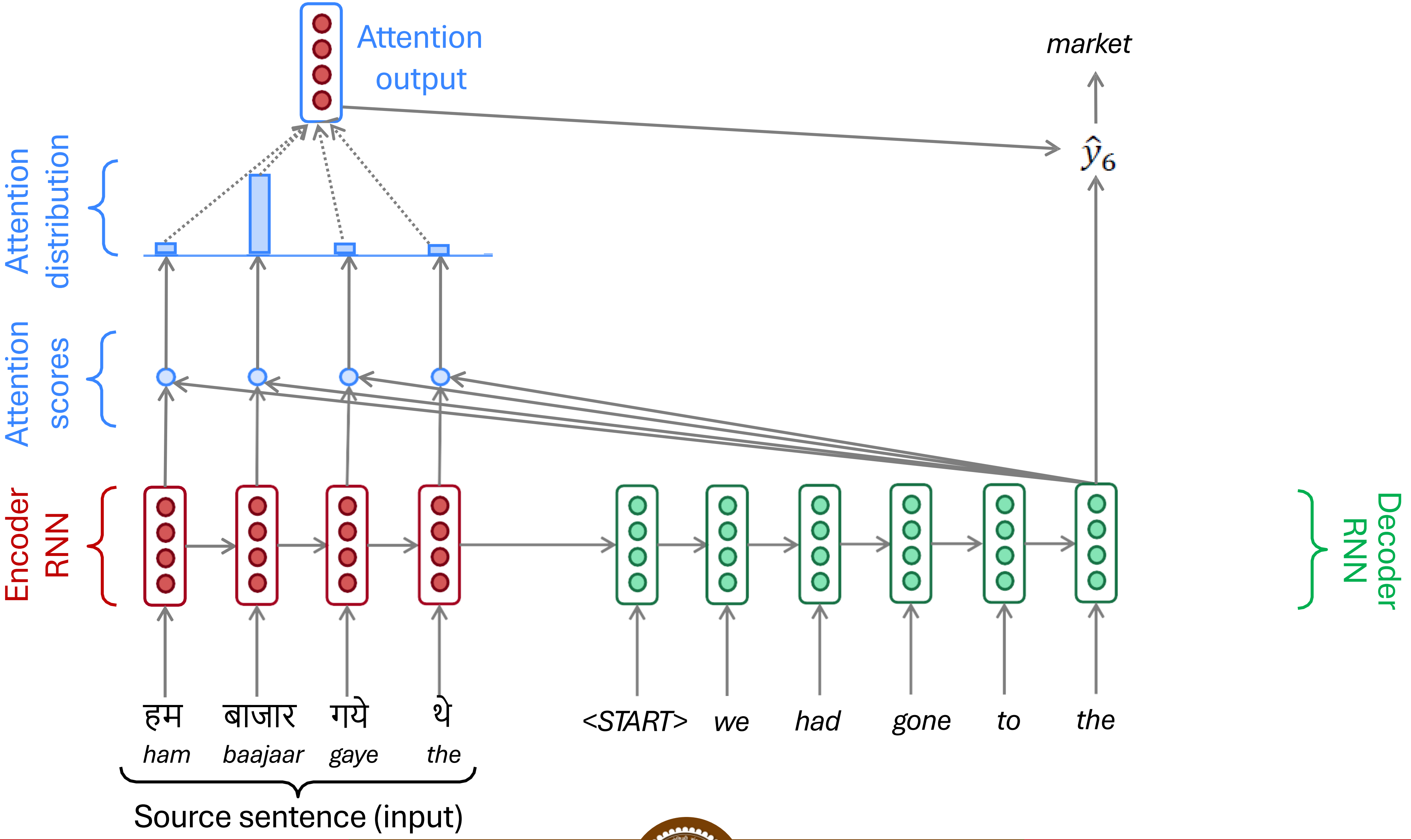
Sequence-to-Sequence With Attention



Sequence-to-Sequence With Attention



Sequence-to-Sequence With Attention



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 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, 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 a_t

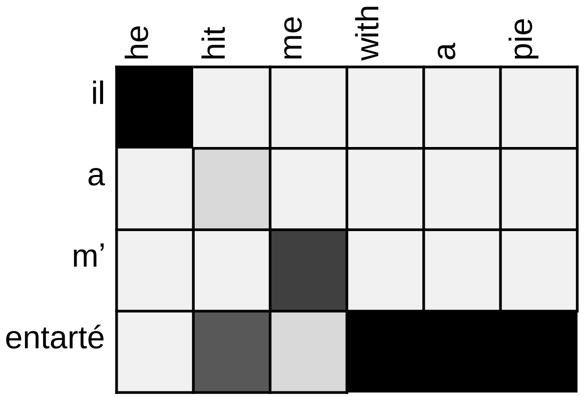
$$\boldsymbol{a}_t = \sum_{i=1}^{N} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

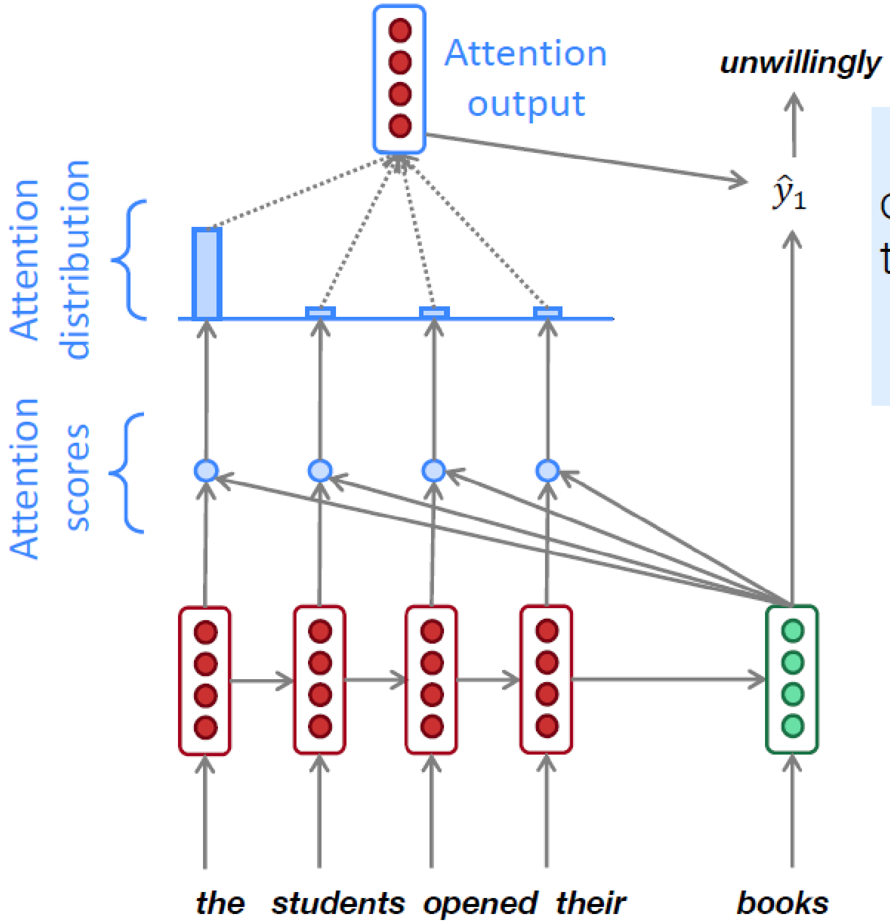
$$[\boldsymbol{a}_t; \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Attention is Great

- Attention significantly improves NMT performance
  - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can 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



Seq2Seq+Attention for LM

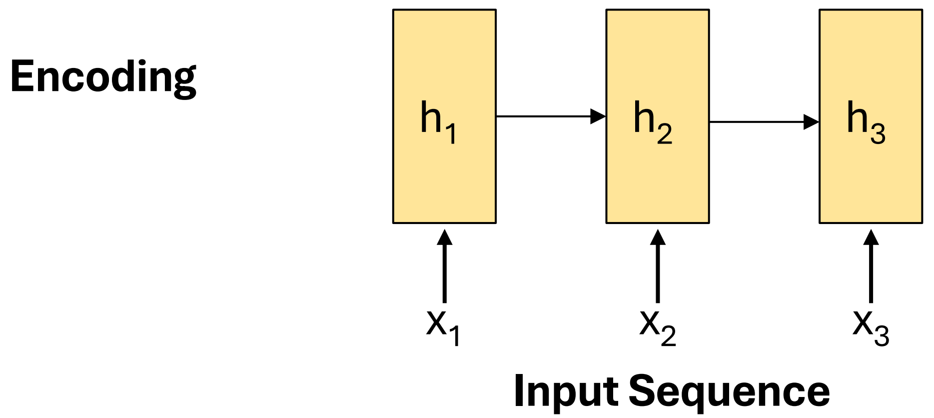


Concatenate (or otherwise compose) the attention output with the current hidden state, then pass through a softmax layer to predict the next word

Attention is a *General* Deep Learning Technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in many architectures (not just seq2seq) and many tasks (not just MT)
- 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.
- We sometimes say that the query *attends to* the values.
- For example, in the seq2seq + attention model, each decoder hidden state (query) *attends to* all the encoder hidden states (values).
- **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).

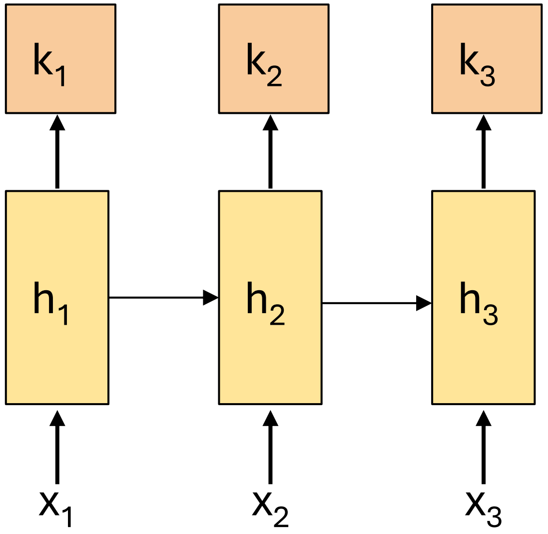
Attention



Attention

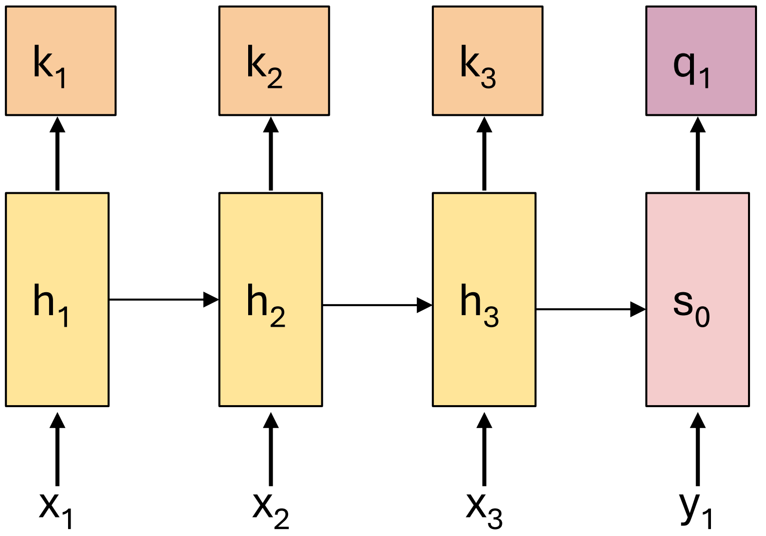
Key vectors represent what **information** is **encoded** at each encoder time step.

Encoding



Input Sequence

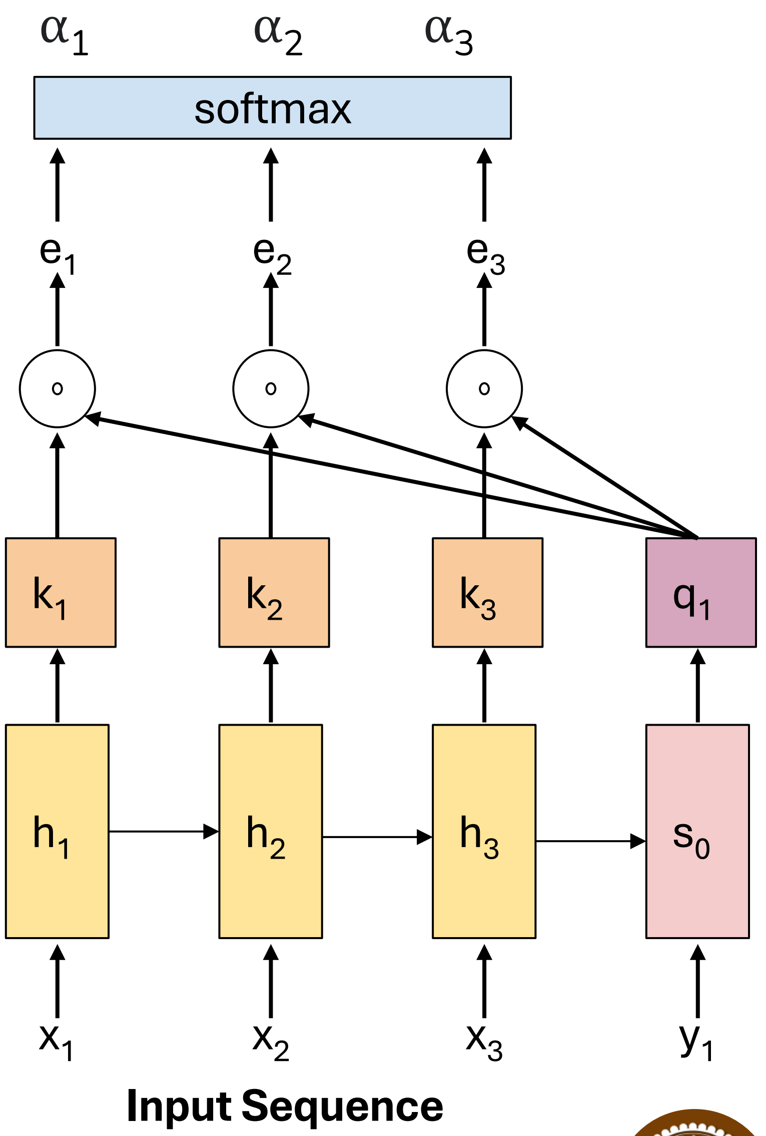
Attention



Query vectors represent what information we are **looking for** at each decoder time step.

Decoding

Attention

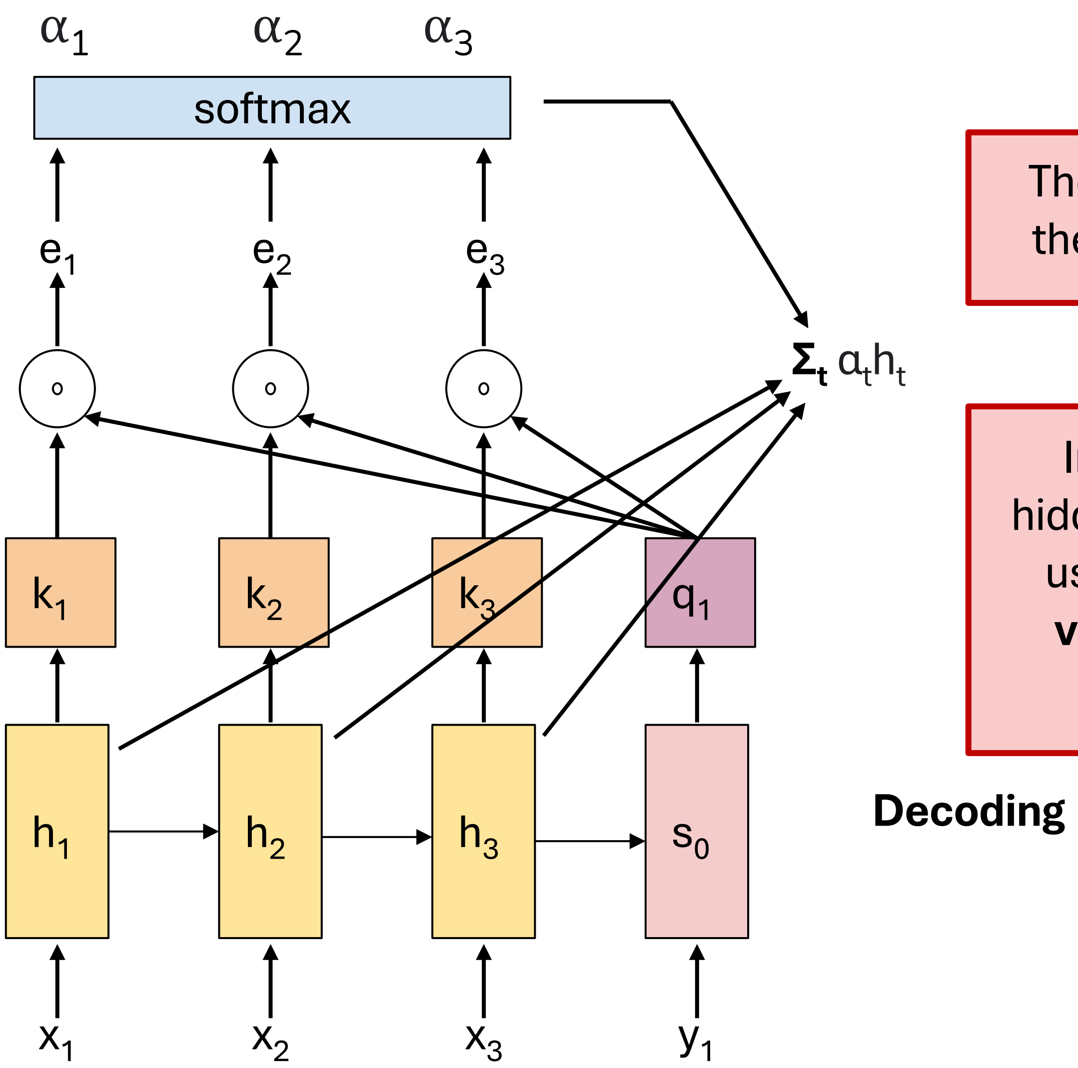


Softmax converts the similarity scores into a **probability distribution.**

Dot product between query vector and every key vector gives **similarity score.**

Decoding

Attention



The output of attention mechanism is the **weighted sum** of hidden vectors.

Instead of simply summing up the hidden vectors, we can transform them using a learned function to generate **value vectors** and then compute a weighted sum.

Variants of Attention

- Original formulation: $a(\mathbf{q}, \mathbf{k}) = w_2^T \tanh(W_1[\mathbf{q}; \mathbf{k}])$
- Bilinear product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T W \mathbf{k}$ — Luong et al., 2015
- Dot product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k}$ — Luong et al., 2015
- Scaled dot product: $a(\mathbf{q}, \mathbf{k}) = \dfrac{\mathbf{q}^T \mathbf{k}}{\sqrt{|\mathbf{k}|}}$ — Vaswani et al., 2017

More information:
"Deep Learning for NLP Best Practices", Ruder, 2017. http://ruder.io/deep-learning-nlp-best-practices/index.html#attention
"Massive Exploration of Neural Machine Translation Architectures", Britz et al, 2017, https://arxiv.org/pdf/1703.03906.pdf

Self-Attention