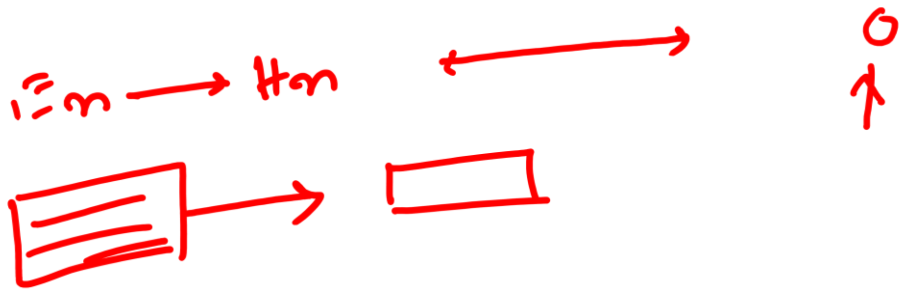
Sequence-to-Sequence Modeling



Qwen3-Coder-Flash Unleashed

High-Performance Code Generation with Agent Integration!

Announced on
August 1, 2025

[Qwen3-Coder]

QWEN3-CODER-FLASH

Qwen3-Coder-30B-A3B-Instruct is a fine-tuned MoE (Mixture-of-Experts) variant in the Qwen3 model family: it has 30.5B total parameters, comprised of 128 experts, with only 8 experts (≈ 3.3B parameters) activated per inference - **making it highly efficient while retaining strong coding and reasoning performance.**

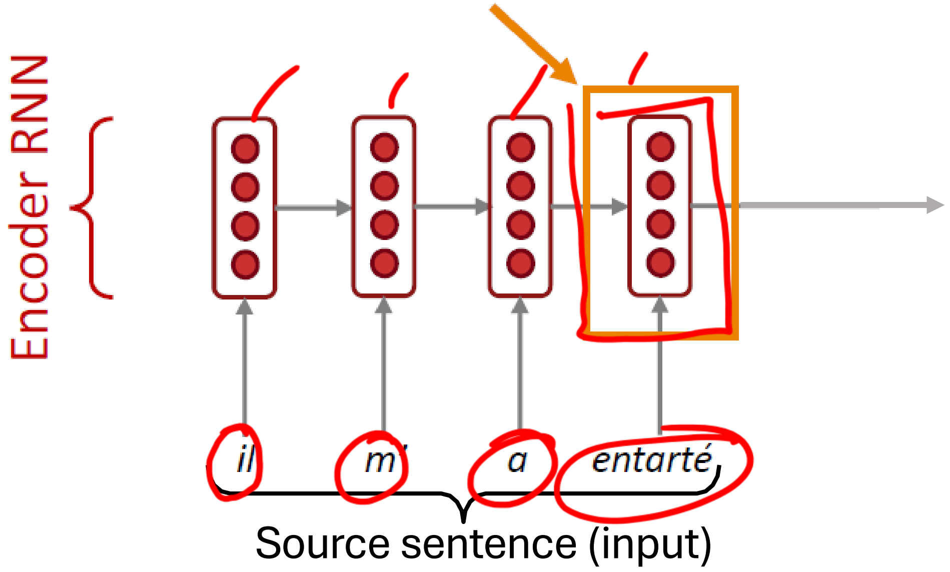
Benchmarks	*Open Models* Qwen3-Coder 30B-A3B-Instruct	Qwen3-Coder 480B-A35B-Instruct	Kimi-K2 Instruct	DeepSeek-V3 0324	*Proprietary Models* Claude Sonnet-4	OpenAI GPT-4.1
Agentic Coding						
Terminal-Bench	31.3	37.5	30.0	2.5	35.5	25.3
SWE-bench Verified						
w/ OpenHands, 500 turns	51.6	69.6	-	-	70.4	-
w/ OpenHands, 100 turns	51.6	67.0	65.4	38.8	68.0	48.6
w/ Private Scaffolding	-	-	65.8	-	72.7	63.8
SWE-bench Live	20.7	26.3	22.3	13.0	27.7	-
SWE-bench Multilingual	34.7	54.7	47.3	13.0	53.3	31.5
Multi-SWE-bench mini	19.5	25.8	19.8	7.5	24.8	-
Multi-SWE-bench flash	19.3	27.0	20.7	-	25.0	-
Aider-Polyglot	33.3	61.8	60.0	56.9	56.4	52.4
Spider2	21.4	31.1	25.2	17.7	31.1	25.6
Agentic Browser Use						
WebArena	43.5	49.9	47.4	40.0	51.1	44.3
Mind2Web	51.0	55.8	42.7	36.0	47.4	49.6
Agentic Tool Use						
BFCL-v3	62.2	68.7	65.2	64.7	73.3	62.9
TAU-Bench Retail	68.7	77.5	70.7	59.1	80.5	-
TAU-Bench Airline	48.0	60.0	53.5	40.0	60.0	-

It shows significant performance among open models on Agentic Coding, Agentic Browser-Use, and other foundational coding tasks. It also features **Long-context Capabilities** with native support for **256K** tokens, extendable up to **1M** tokens using Yarn, optimized for repository-scale understanding.

Neural Machine Translation (NMT)

The Sequence-to-Sequence Model

Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.

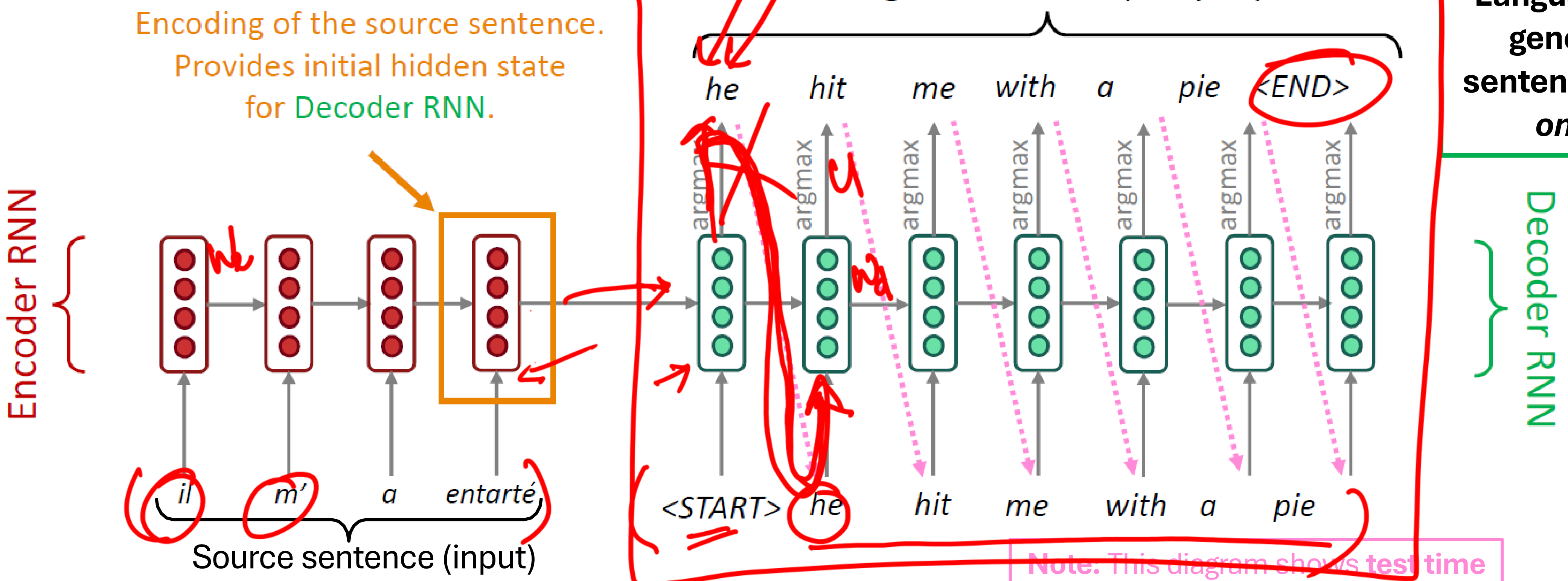


Encoder ~~RNN produces an encoding~~ of the source sentence.

Techer-forcing

Neural Machine Translation (NMT)

The Sequence-to-Sequence Model



Decoder RNN is a **Language Model that generates target sentence, *conditioned on encoding*.**

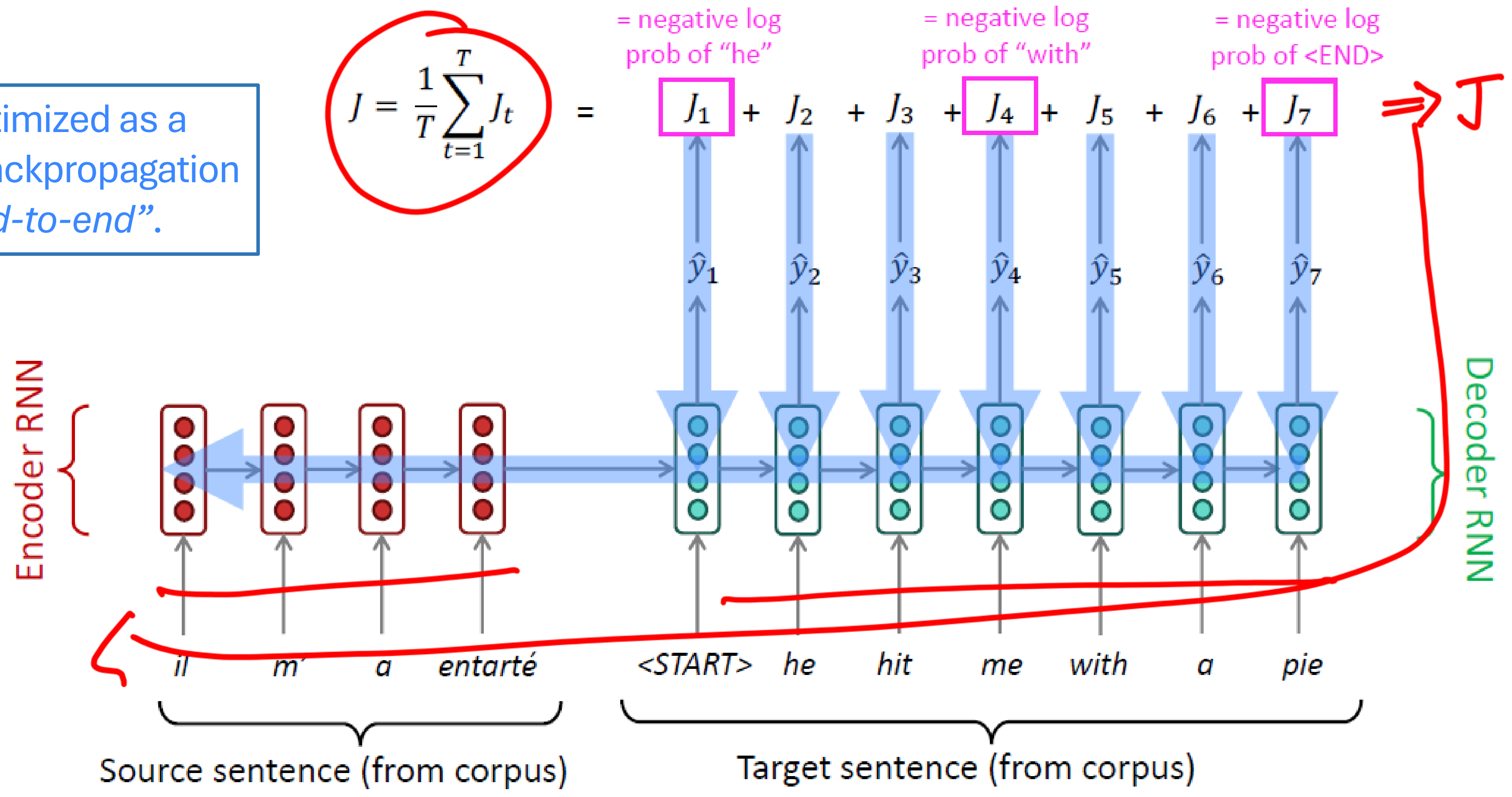
Encoder RNN produces an encoding of the source sentence.

Note: This diagram shows test time behavior: decoder output is fed in as next step's input

Training an NMT System

Seq2seq is optimized as a **single system.** Backpropagation operates "*end-to-end*".

$$J = \frac{1}{T}\sum_{t=1}^{T} J_t = \boxed{J_1} + J_2 + J_3 + \boxed{J_4} + J_5 + J_6 + \boxed{J_7} \Rightarrow J$$

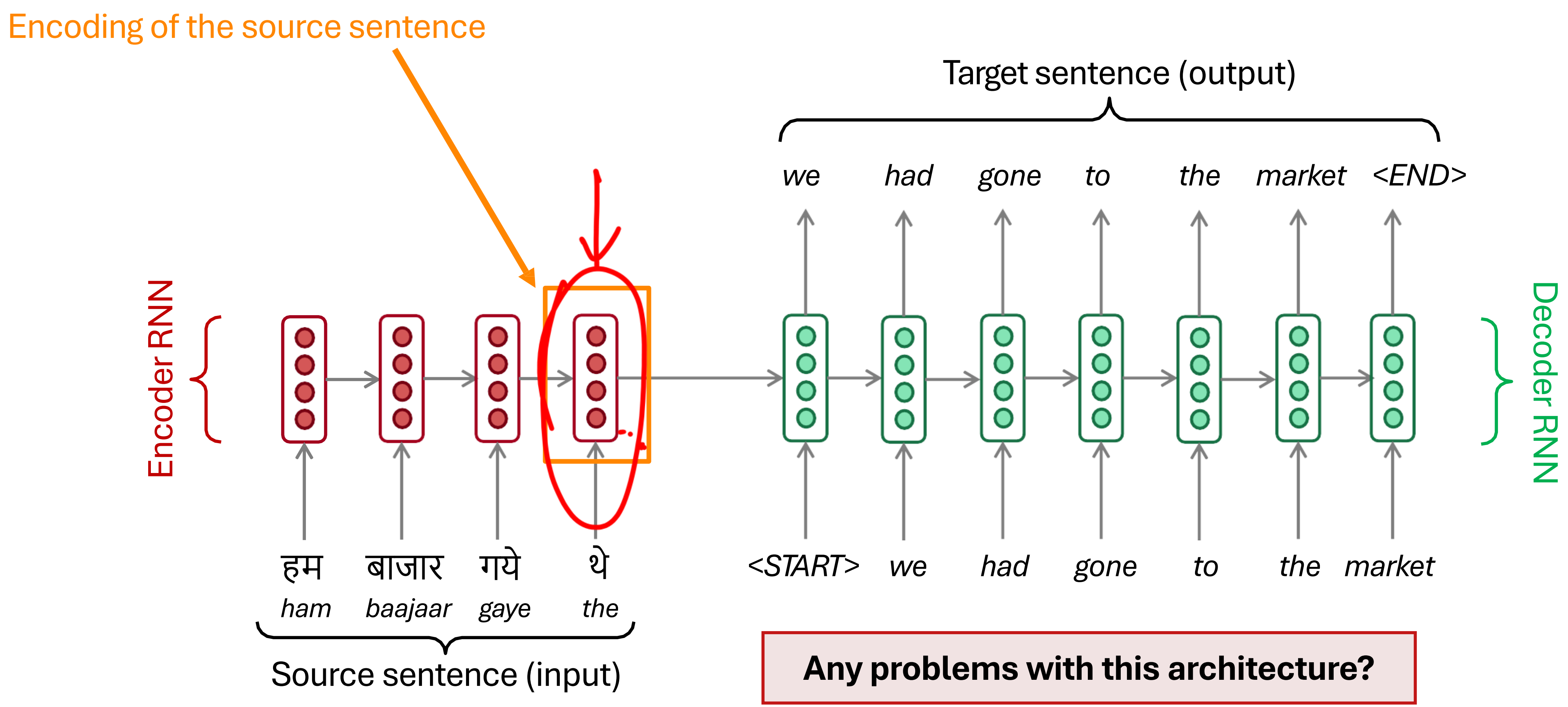


Issues With RNN

- Linear interaction distance
- Bottleneck problem
- Lack of parallelizability

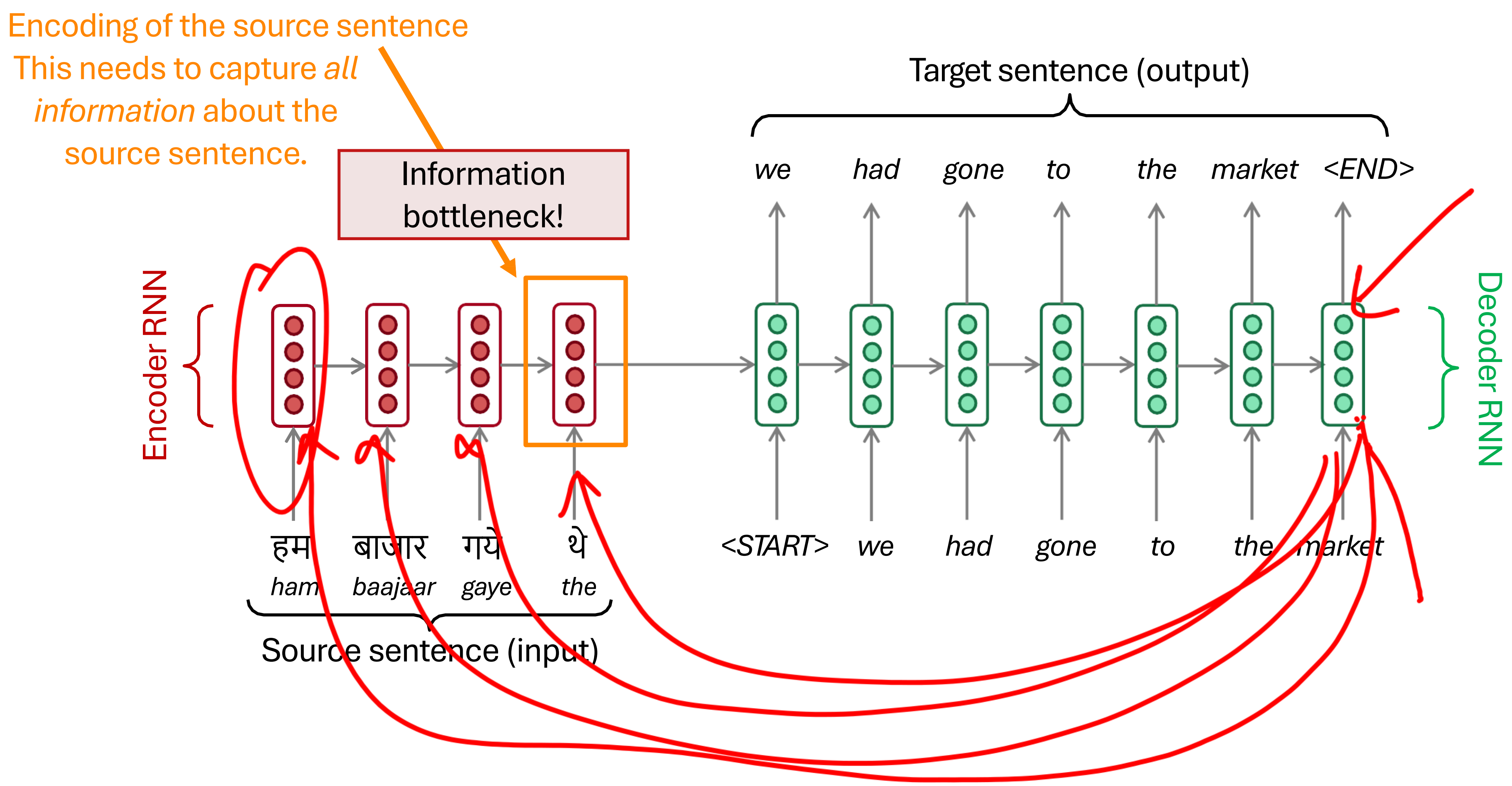
ATTENTION

Sequence-to-Sequence: The Bottleneck Problem

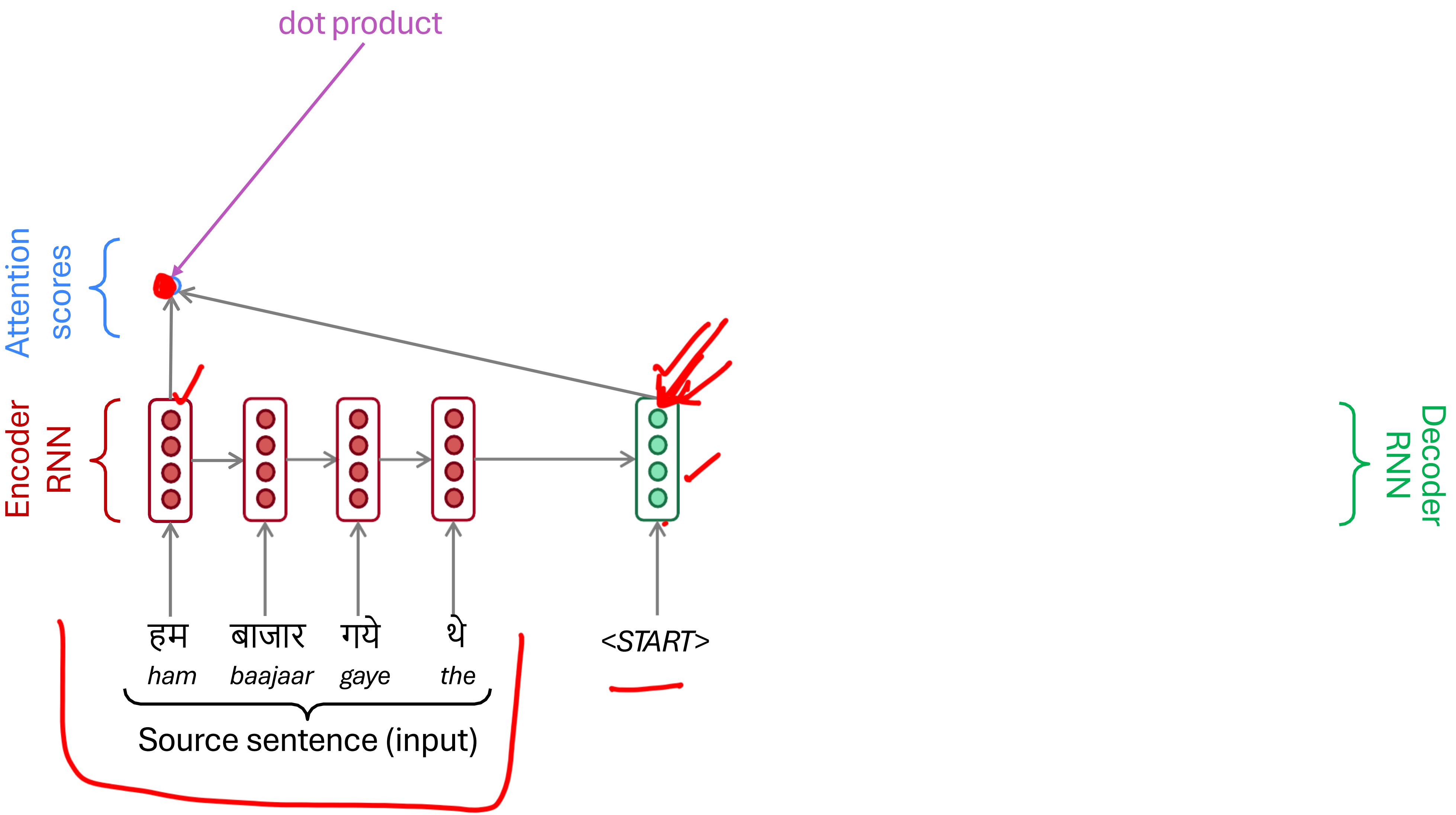


Any problems with this architecture?

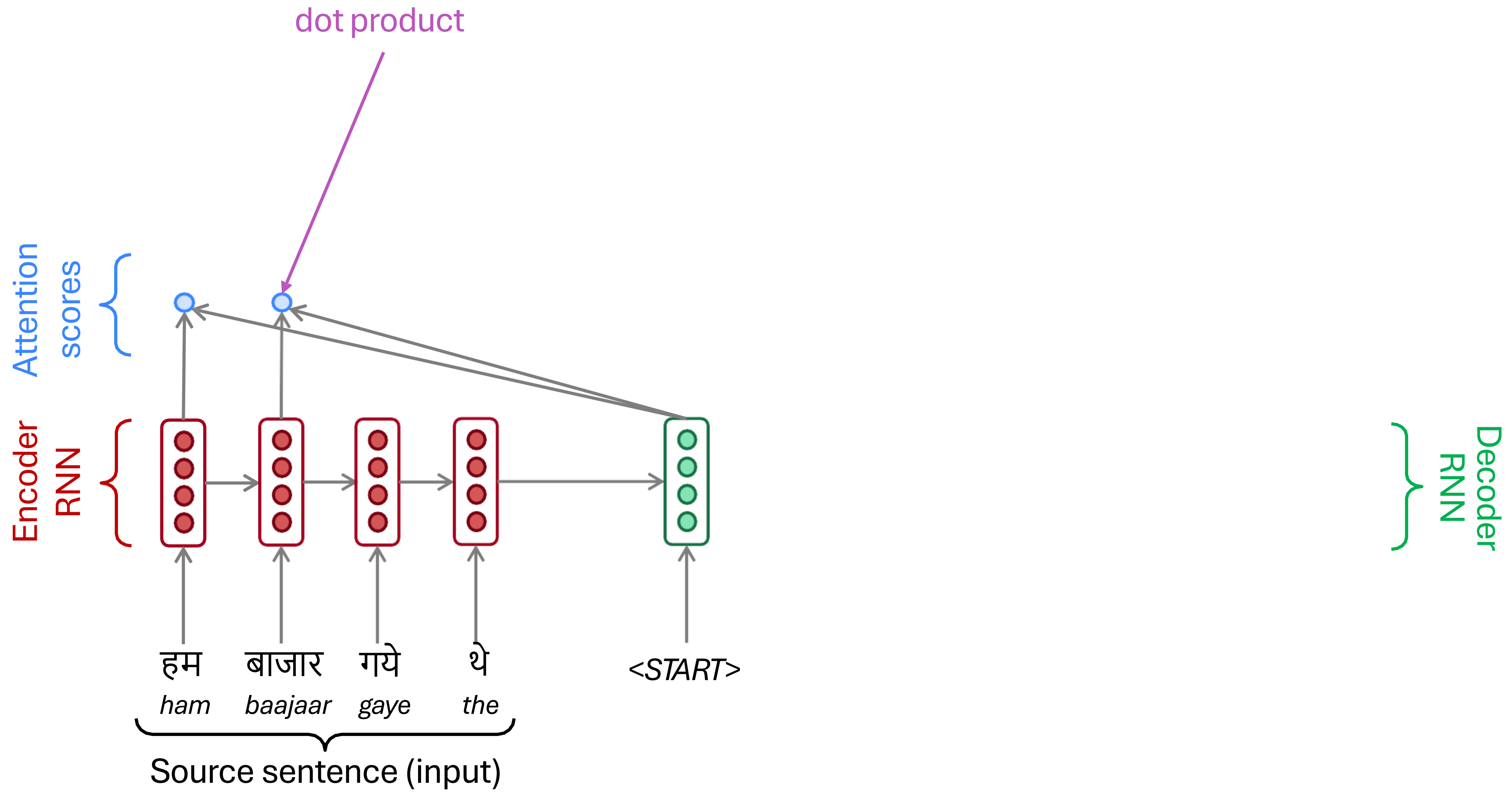
Sequence-to-Sequence: The Bottleneck Problem



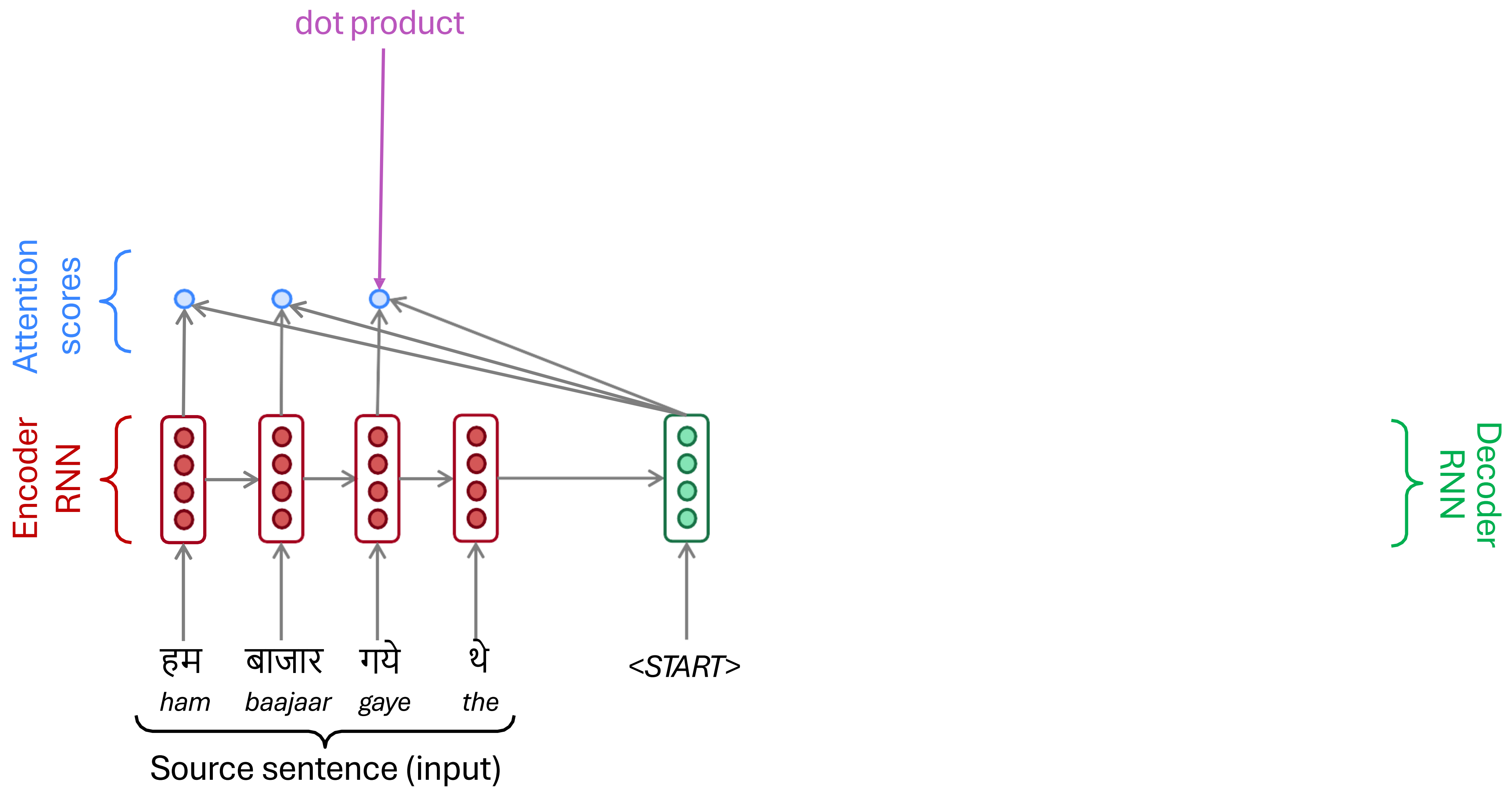
Sequence-to-Sequence With Attention



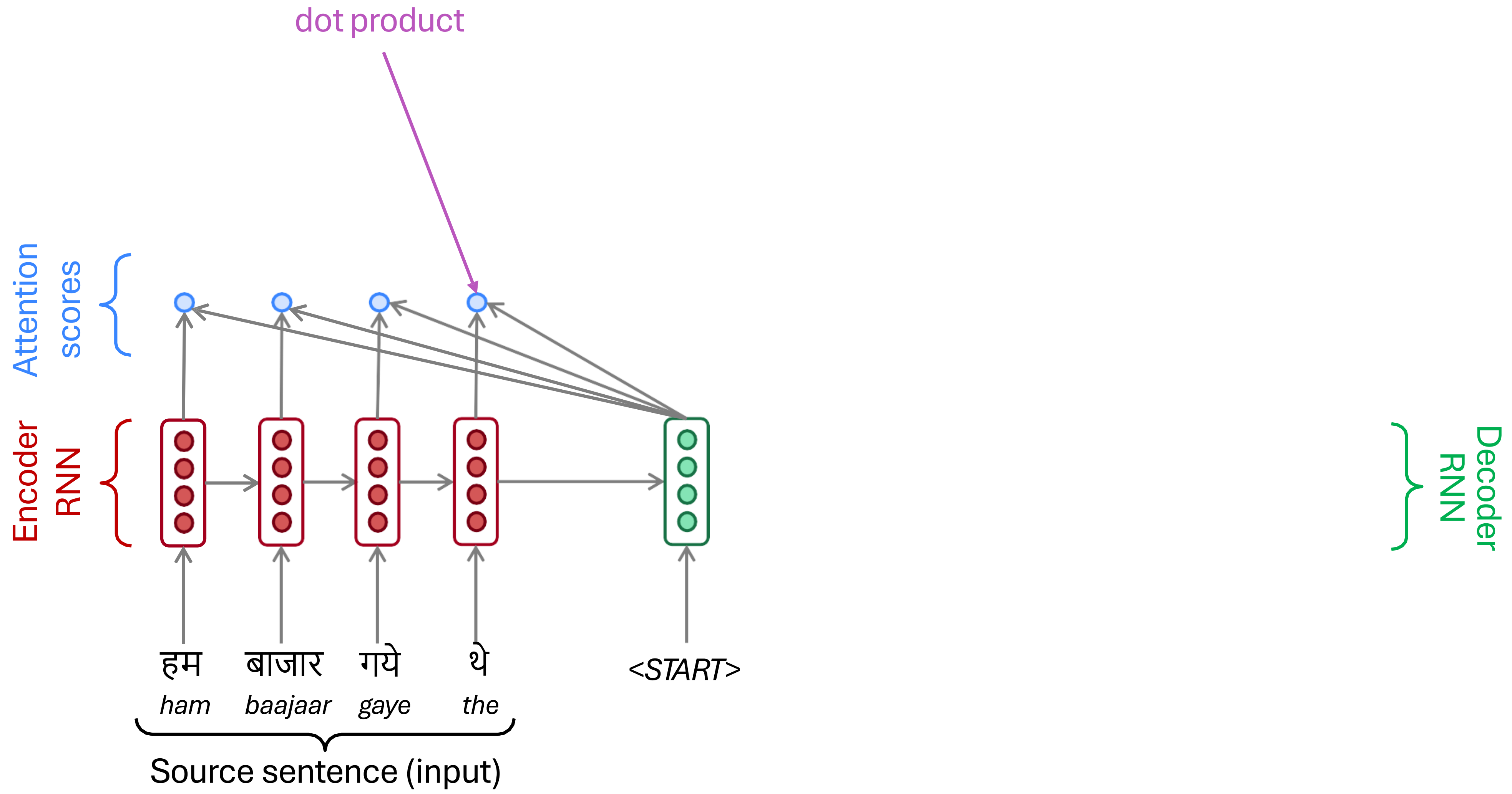
Sequence-to-Sequence With Attention



Sequence-to-Sequence With Attention



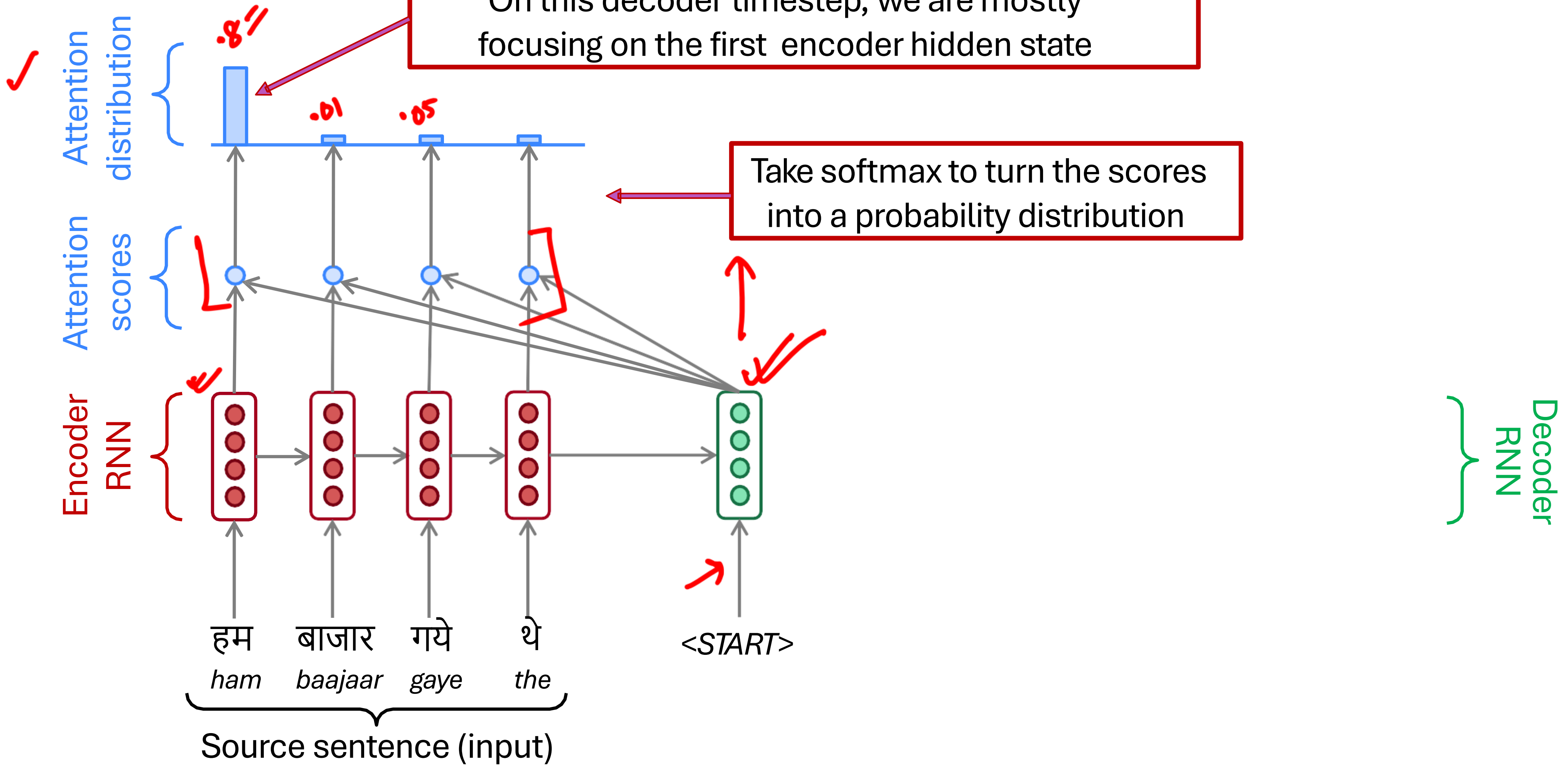
Sequence-to-Sequence With Attention



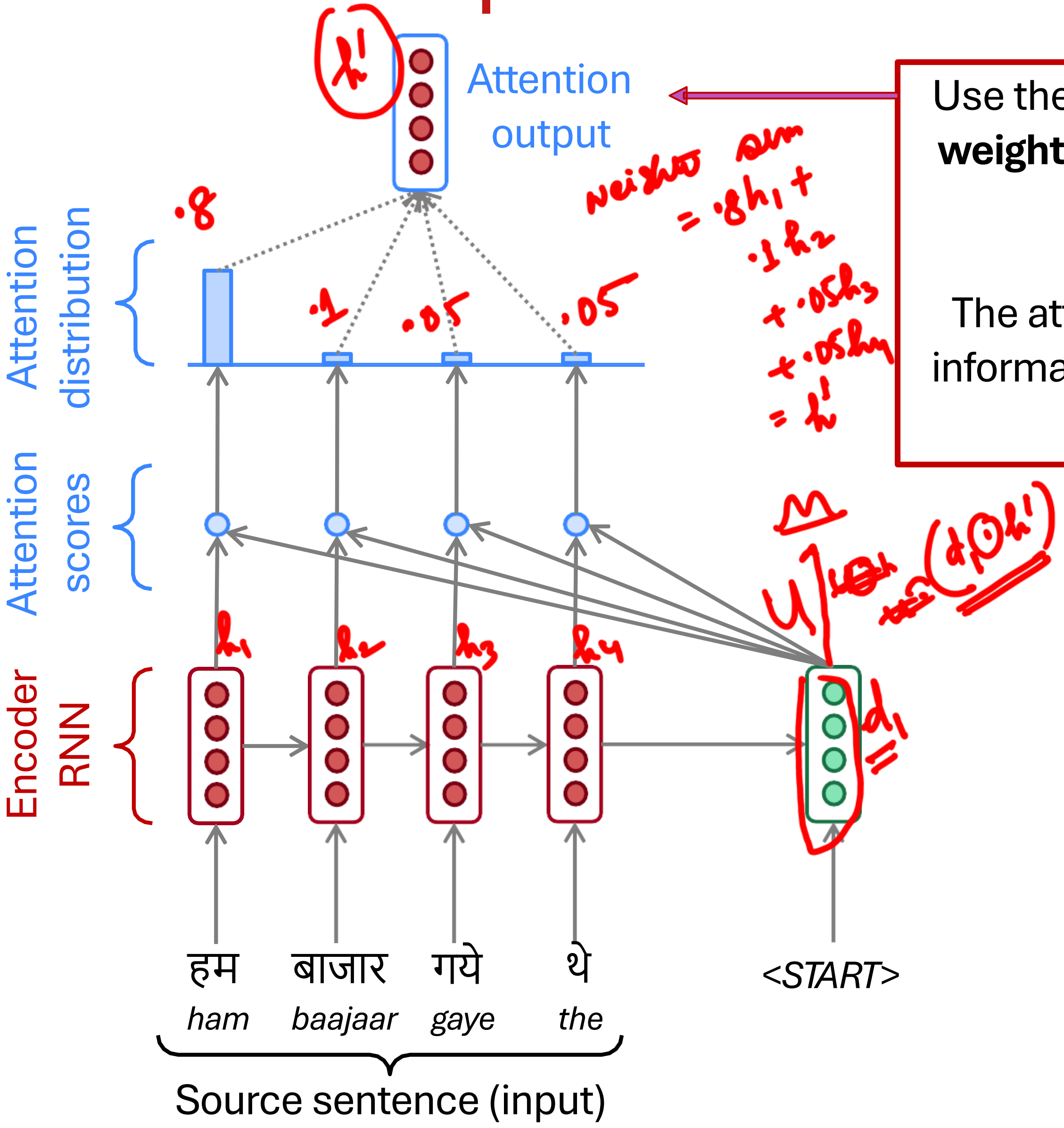
Sequence-to-Sequence With Attention

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

Take softmax to turn the scores into a probability distribution



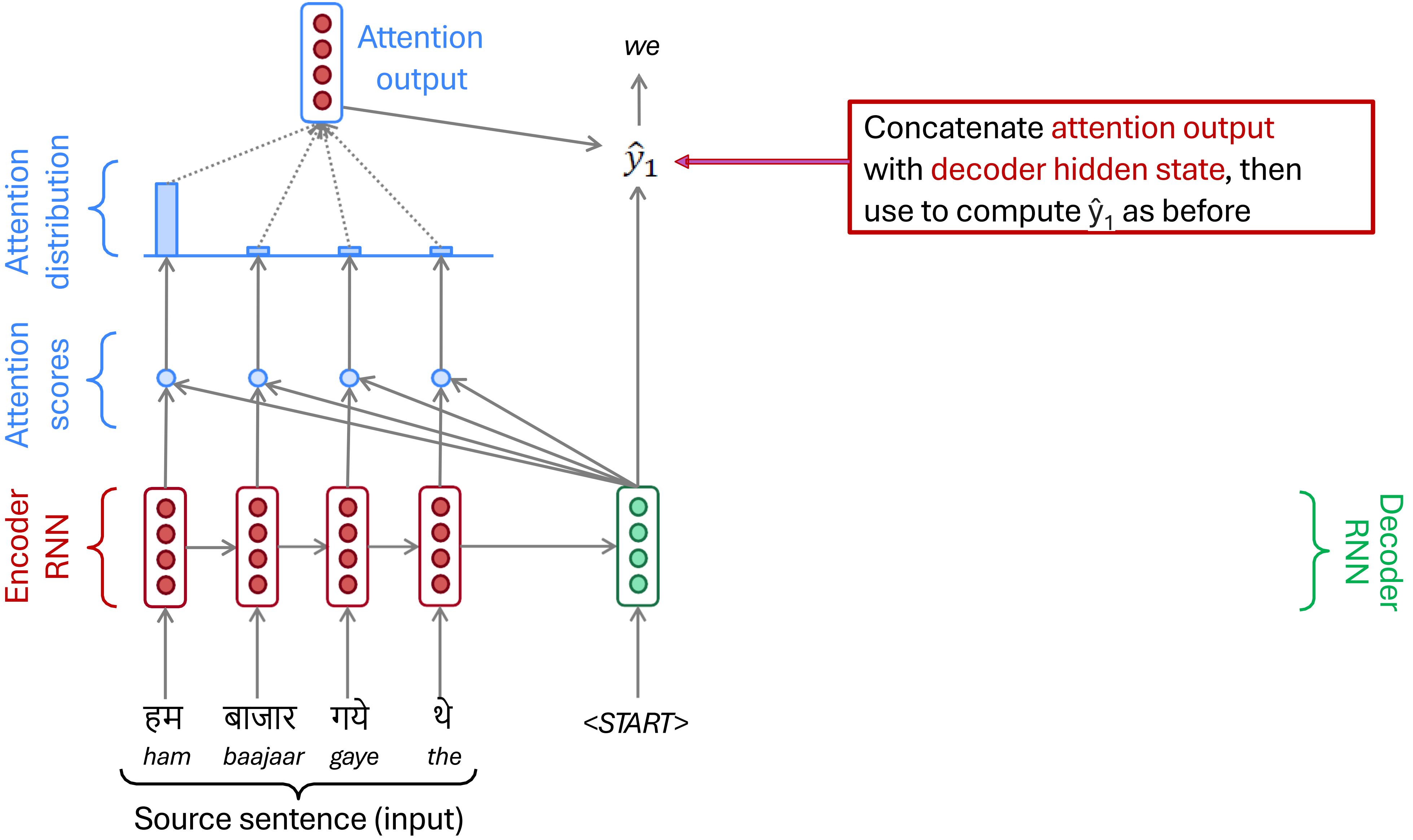
Sequence-to-Sequence With Attention



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

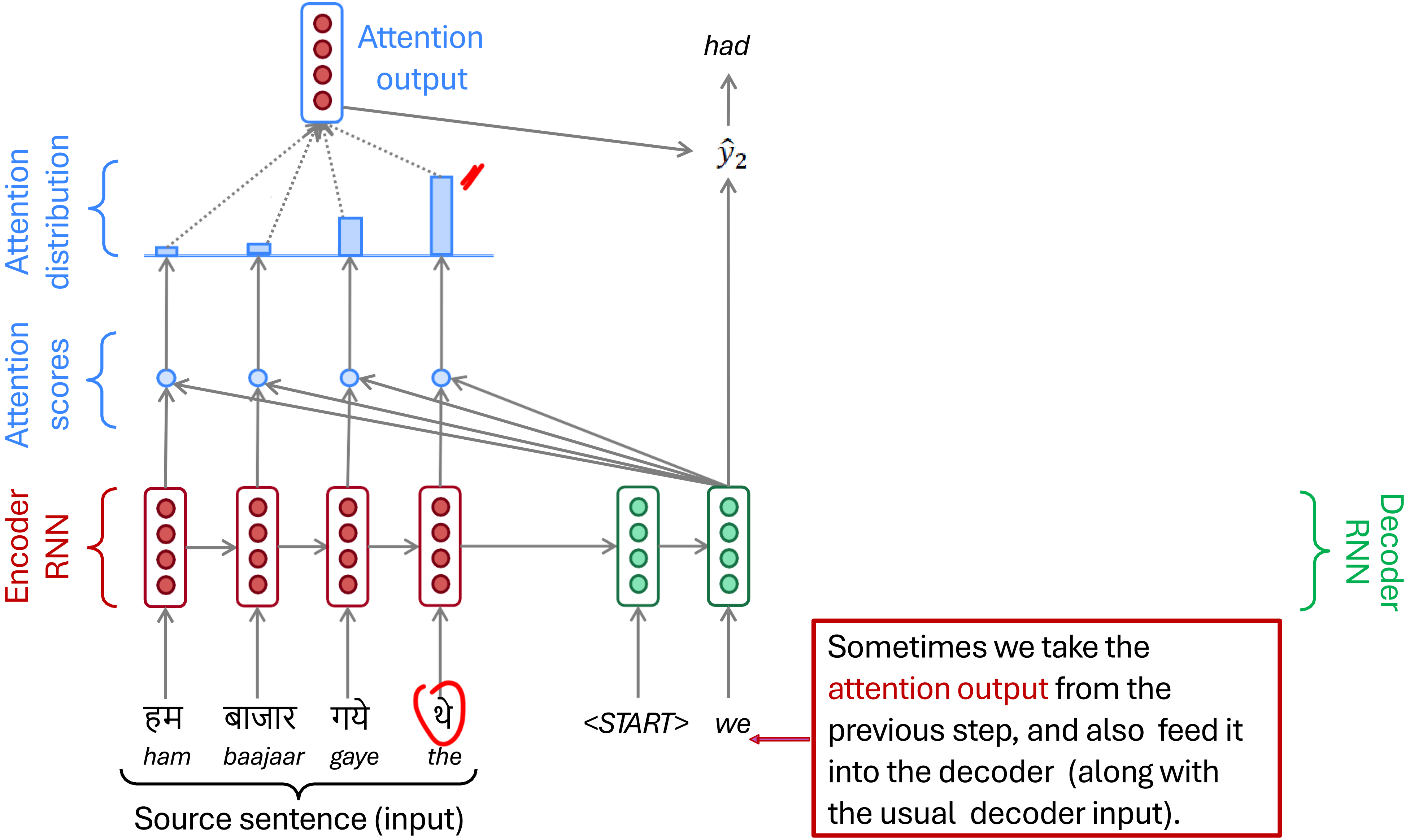
The attention output mostly contains information from the hidden states that received high attention.

Sequence-to-Sequence With Attention



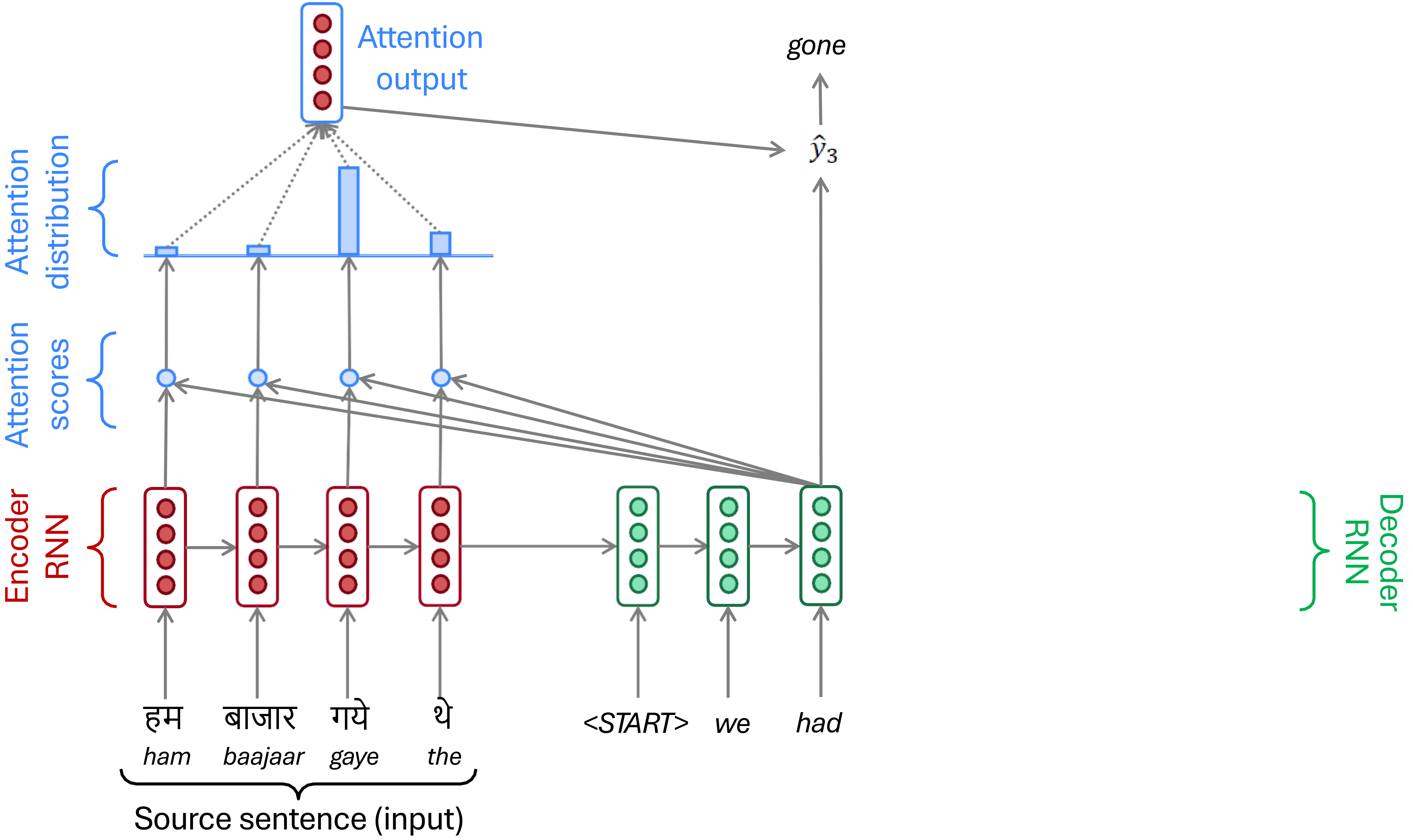
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Sequence-to-Sequence With Attention

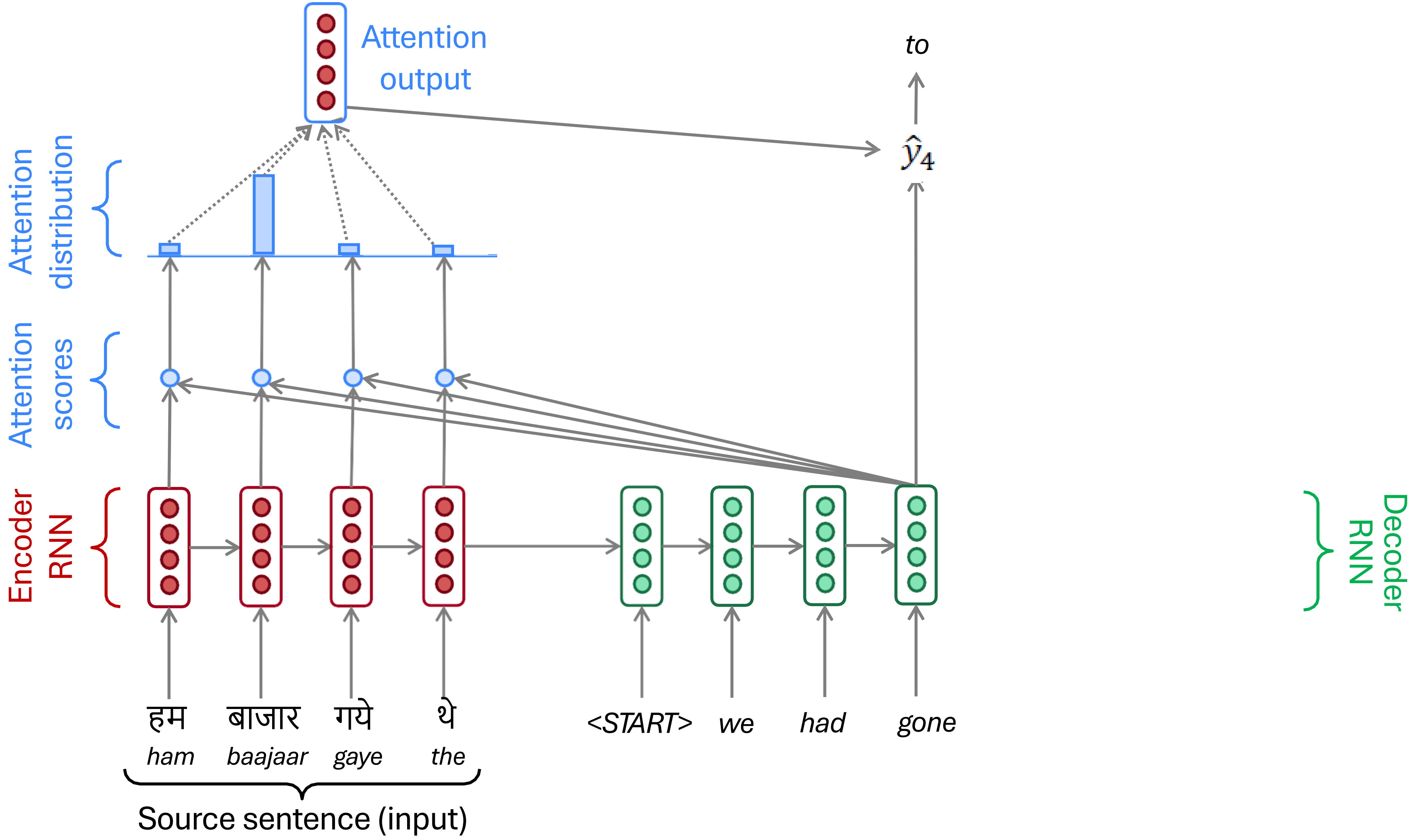


Sometimes we take the attention output from the previous step, and also feed it into the decoder (along with the usual decoder input).

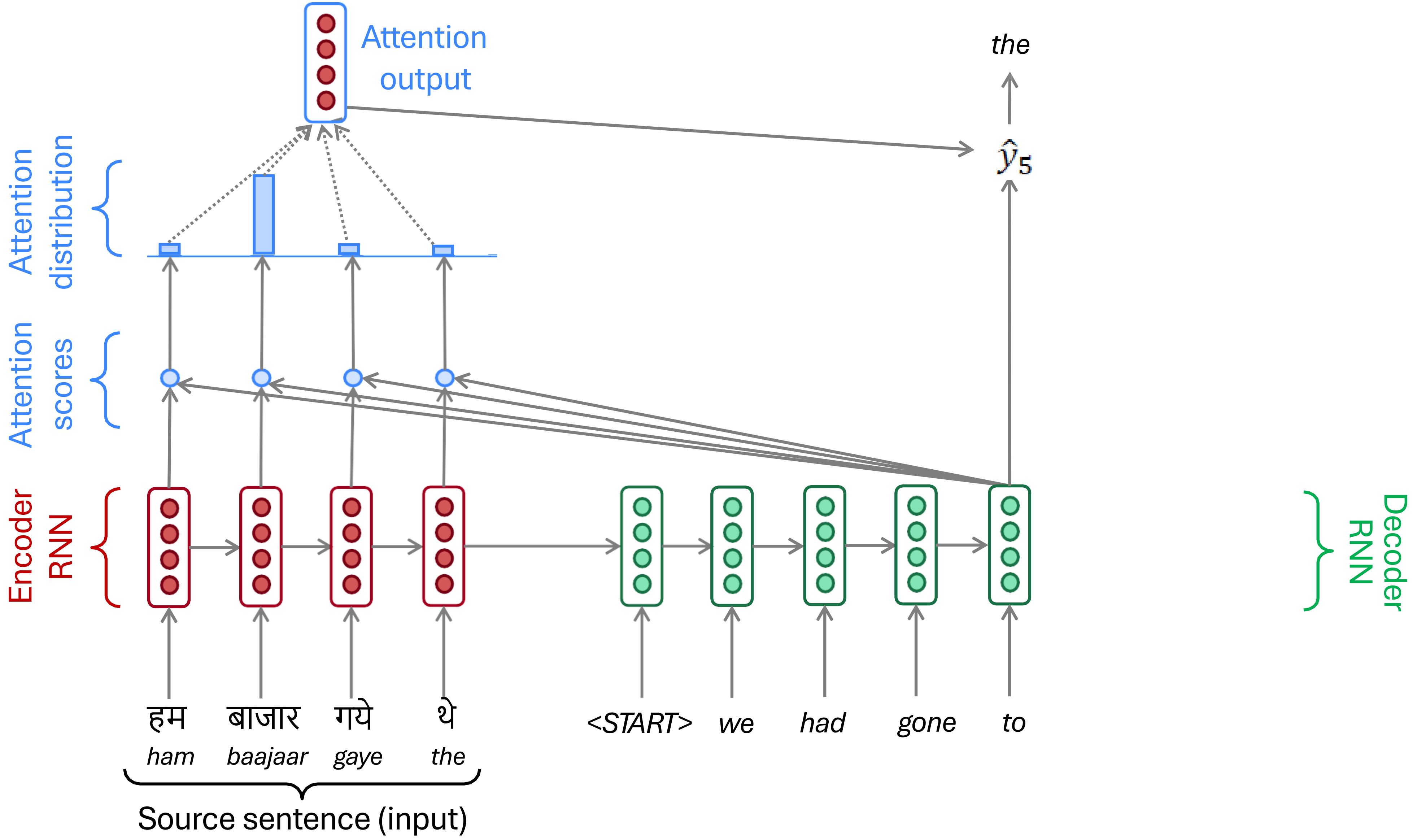
Sequence-to-Sequence With Attention



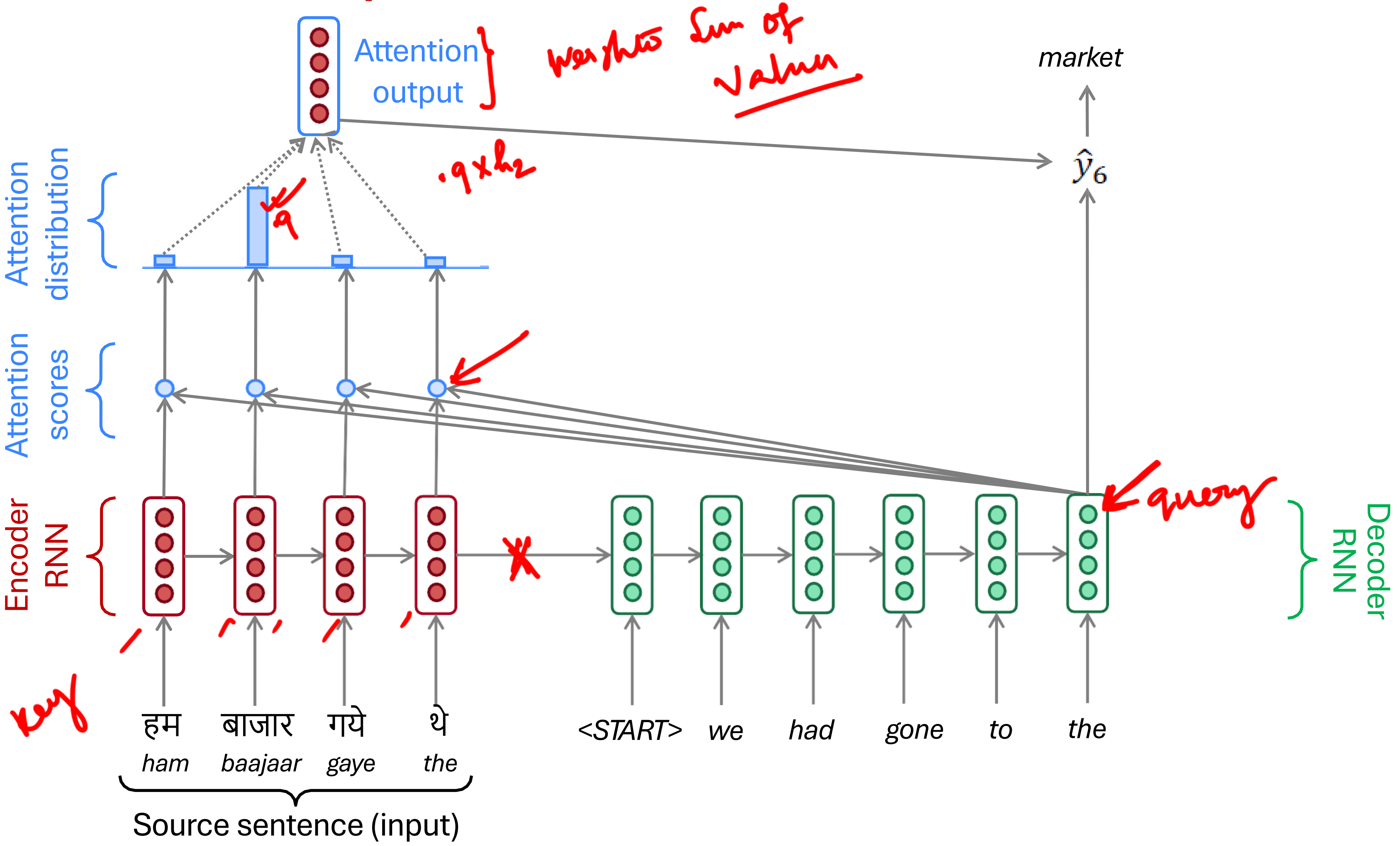
Sequence-to-Sequence With Attention



Sequence-to-Sequence With Attention

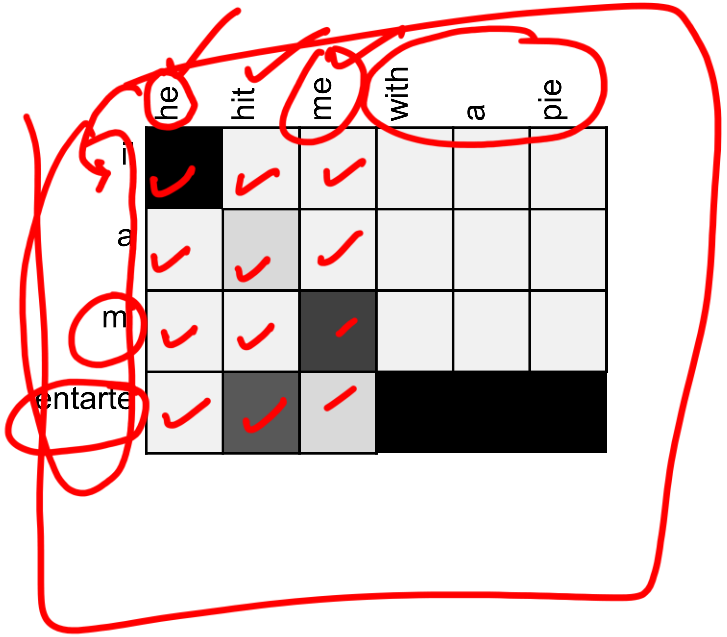


Sequence-to-Sequence With Attention



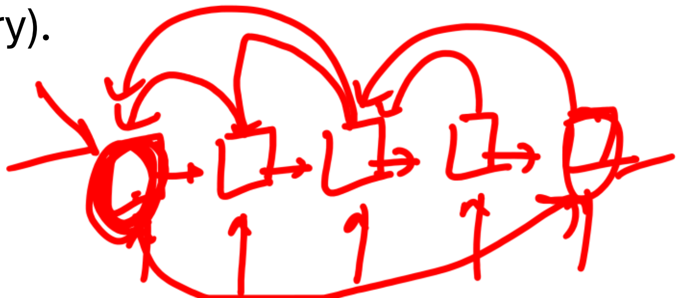
Attention is Great

- 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



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.
- <u>However</u>: 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).



Variants of Attention

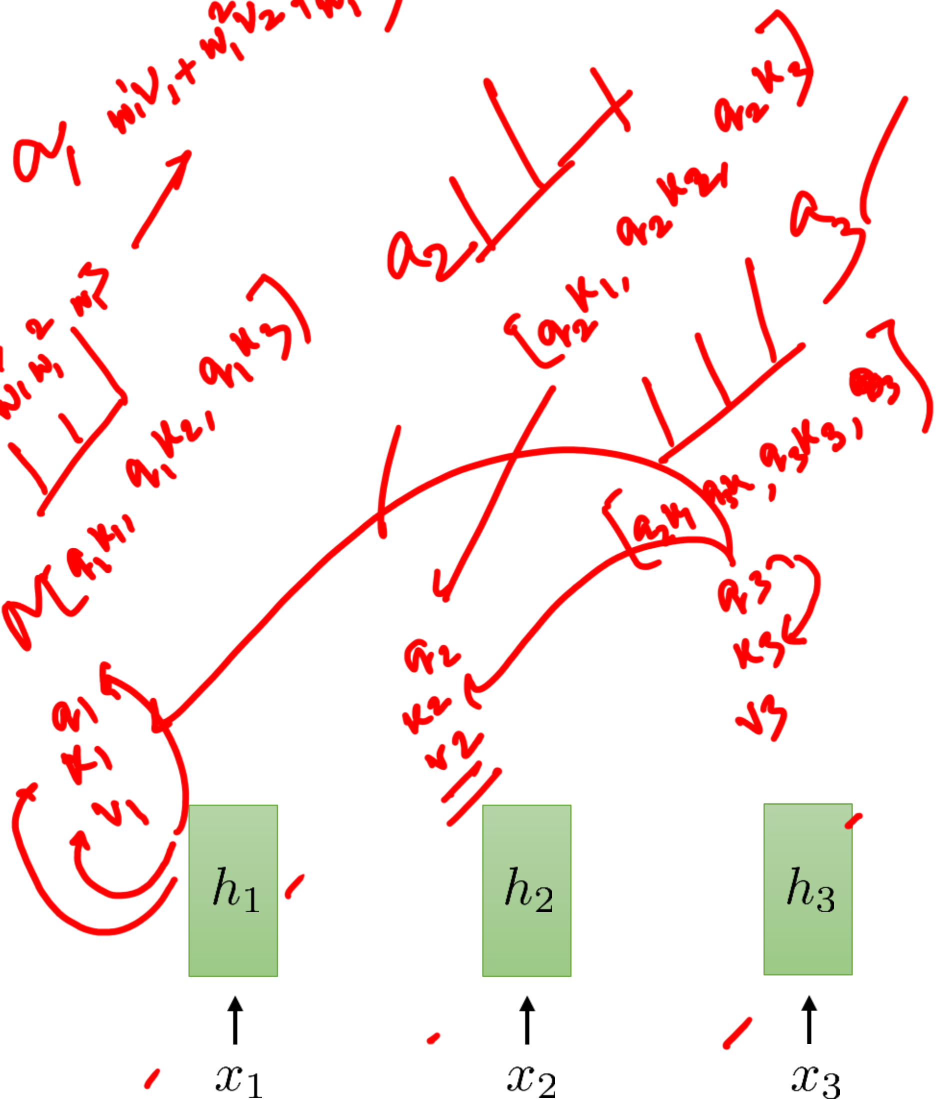
- Original formulation: $a(\mathbf{q}, \mathbf{k}) = w_2^T \tanh(W_1[\mathbf{q}; \mathbf{k}])$
- Bilinear product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T W \mathbf{k}$ Luong et al., 2015
- Dot product: $a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T \mathbf{k}$ Luong et al., 2015
- Scaled dot product: $a(\mathbf{q},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

W_q
W_k
W_v

$q_1 = W_q h_1$
$k_1 = W_k h_1$
$v_1 = W_v h_1$



this is *not* a recurrent model!
but still weight sharing:

$$h_t = \sigma(Wx_t + b)$$

shared weights at all time steps

(or any other nonlinear function)