SCALER School of Technology × Priyansh Saxena

Sequence-to-Sequence Models and Attention

Encoder-Decoder • Bottleneck • Bahdanau • Luong • QKV

Priyansh Saxena

Scaler School of Technology | Deep Learning I: Neural Networks | Week 6 — Session 2/2

6 June 2026

The Translation Revolution

Before (2014):

Phrase-based statistical MT
Choppy, ungrammatical output
“The cat it is on the mat of the”

After (2016):

Neural Machine Translation
Fluent, natural output
“The cat is on the mat”

What changed? Encoder-decoder architecture + **attention mechanism.**

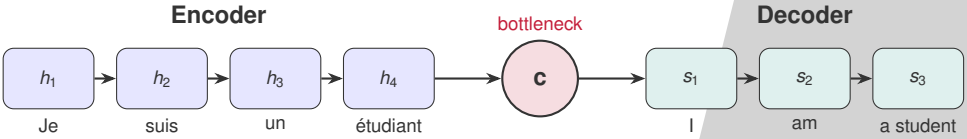
The Problem: Sequence $\rightarrow$ Sequence

Challenge

- Input: variable-length sequence (French sentence)
- Output: variable-length sequence (English sentence)
- Different lengths, different word orders
- One-to-many and many-to-one word mappings

We need: An architecture that handles *any* length in $\rightarrow$ *any* length out.

Encoder-Decoder Architecture



Key idea: Compress the entire input into a single vector **c**, then decode from it.

The Encoder

- Processes input tokens left-to-right (one per timestep)
- $h_t = \text{RNN}(x_t, h_{t-1})$
- Final hidden state h_T = “summary” of the entire input
- Uses LSTM or GRU (from last session!)

The encoder produces no output — it just builds up a representation.

Think of it as reading a sentence and holding the meaning in your head.

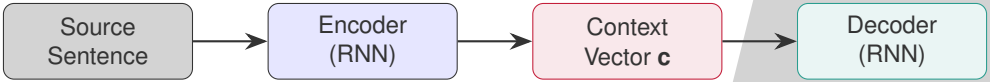
The Decoder

- Receives context vector **c** as initial hidden state
- Generates one token at a time: $y_1, y_2, \ldots$ until $\langle \text{END} \rangle$
- **Autoregressive:** each output depends on all previous outputs

Teacher Forcing

During training: feed the **correct** previous token (not the predicted one).
At inference: feed your own predictions (no ground truth available).

The Full Seq2Seq Pipeline



Loss = Cross-Entropy on output tokens

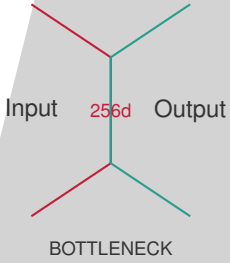
Sutskever et al., 2014: *"Sequence to Sequence Learning with Neural Networks"*

It works! . . . but there's a problem.

The Information Bottleneck Problem

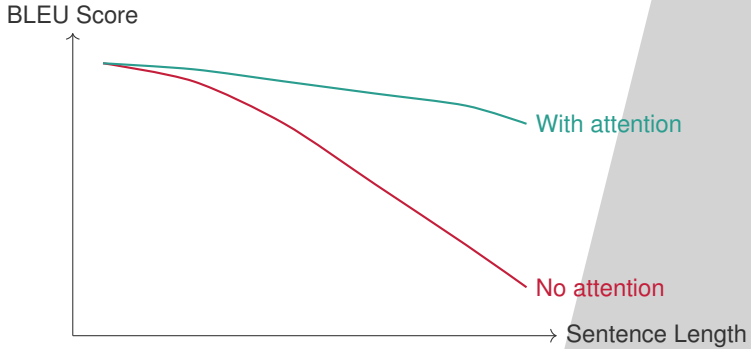
The Problem

- Entire input $\rightarrow$ ONE vector (e.g., 256-dim)
- Short sequences: fits fine
- Long sequences: information is lost
- Early words “forgotten” by the time encoding finishes



“Summarize a 50-page book into a Post-it note, then recreate the book from it.”

Evidence: BLEU vs. Sentence Length



The longer the input, the worse basic seq2seq performs.
Attention keeps quality high regardless of length.

Quick Check

Question

Q1: In the basic encoder-decoder, what information does the decoder have access to from the input?

Q2: Why does translation quality degrade for longer sentences?

Quick Check

Question

Q1: In the basic encoder-decoder, what information does the decoder have access to from the input?

Q2: Why does translation quality degrade for longer sentences?

A1: Only the final hidden state — one fixed-size context vector.
A2: Fixed capacity can't hold all info; early words are forgotten.

Coming Up Next

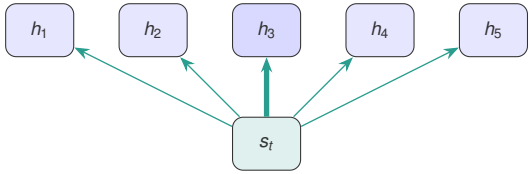
Part 2: The Attention Mechanism — Solving the Bottleneck

Attention: The Key Insight

Key Idea

Instead of one context vector for the *entire* sentence. . .

Give the decoder access to **ALL** encoder hidden states.
Let it **choose** which states matter at each decoding step.



Open-book exam instead of closed-book.

Attention: Step by Step

At each decoder step t:

1. **Score:** $e_{ti} = \text{score}(s_t, h_i)$ for each encoder state h_i
2. **Normalize:** $\alpha_{ti} = \text{softmax}(e_{ti})$ — attention weights
3. **Context:** $c_t = \sum_i \alpha_{ti} \cdot h_i$ — weighted sum
4. **Predict:** use $[c_t; s_t]$ to generate output token

Key: A fresh context vector c_t at EVERY decoding step.
No more single bottleneck!

Bahdanau (Additive) Attention

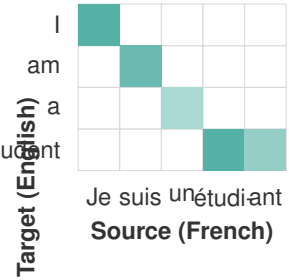
Scoring Function

$$\text{score}(s, h) = \mathbf{v}^\top \cdot \tanh(\mathbf{W}_s \cdot s + \mathbf{W}_h \cdot h)$$

- $\mathbf{W}_s$, $\mathbf{W}_h$, $\mathbf{v}$ are **learnable** parameters
- It's a tiny neural network (1 hidden layer)
- Called "additive" because it *adds* transformed states
- The network learns what makes two states relevant to each other

Bahdanau et al., 2015: *"Neural Machine Translation by Jointly Learning to Align and Translate"*

Attention Weights: Alignment Heatmap



The model learns word alignment automatically — no supervision needed!
Non-diagonal = word reordering between languages.

Impact: Before vs. After Attention

	Without Attention	With Attention
Context	Single fixed vector	Fresh vector per step
Long sequences	Degrades badly	Stays strong
Interpretability	Black box	Can visualize alignment
BLEU (long)	Drops sharply	Maintains quality

**Attention doesn’t just improve accuracy —
it makes the model interpretable.**

Luong Attention: Simpler Variants

Name	Formula	Parameters
Dot-product	$\text{score} = s^\top h$	None
General	$\text{score} = s^\top W h$	W matrix
Concat (= Bahdanau)	$\text{score} = v^\top \tanh(W[s; h])$	W, v

Luong et al., 2015: showed simpler variants work *nearly* as well.

Dot-product: fastest, no extra parameters, just a matrix multiply.

Dot-Product vs. Additive Attention

Dot-Product

- No extra parameters
- Very fast (matrix multiply)
- Scales well to large dims
- Used in Transformers

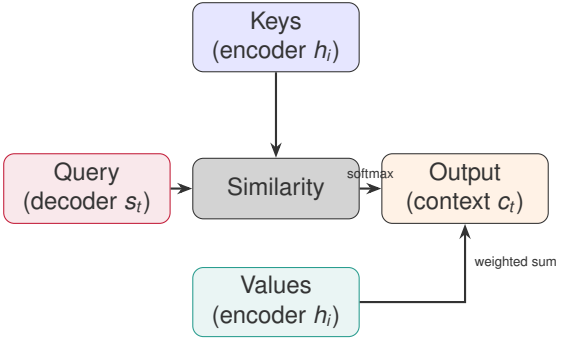
Additive (Bahdanau)

- Learnable parameters
- More expressive
- Slower (extra layer)
- Better for small dims

"In practice, dot-product wins for modern model sizes."

Preview: Transformers use **scaled** dot-product: $\frac{s^\top h}{\sqrt{d_k}}$

Attention as Query-Key-Value Lookup



- **Query:** what am I looking for?
- **Keys:** what does each position advertise?
- **Values:** what gets retrieved?

$$\text{Attn} = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Hard vs. Soft Lookup

Hard Lookup:

- Find the ONE best match (argmax)
- Like a hash table
- Not differentiable!

Soft Lookup (Attention):

- Weighted combination of ALL values
- Like a fuzzy search
- Fully differentiable!

Because it's soft, we can train it end-to-end with backprop.

No alignment supervision needed — it learns from the translation objective alone.

Applications Beyond Translation

- **Text Summarization**
  Attend to key sentences
- **Image Captioning**
  Attend to image regions
- **Speech Recognition**
  Attend to audio frames
- **Question Answering**
  Attend to relevant passages
- **Code Generation**
  Attend to documentation
- **Music Transcription**
  Attend to audio segments

Anywhere you need to select relevant info from a sequence $\rightarrow$ attention works.

Quick Check

Question

Q1: In dot-product attention, what determines which encoder states get high weight?

Q2: How does attention solve the information bottleneck?

Quick Check

Question

Q1: In dot-product attention, what determines which encoder states get high weight?

Q2: How does attention solve the information bottleneck?

A1: Similarity (dot product) between decoder state and encoder state.
A2: Fresh context vector at each step from ALL encoder states — no single bottleneck.

Recap: The Story Arc

1. **Encoder-Decoder** — read input → one vector → generate output
2. **Bottleneck Problem** — one vector can't hold everything
3. **Attention** — let decoder look back at all positions
4. **Variants** — Bahdanau (additive) vs. Luong (dot-product)
5. **QKV Framing** — query searches keys, retrieves values

This entire progression leads to one question. . .

Next: The Transformer

What if we removed the RNN entirely?

What if attention was the **only** mechanism?

What if we applied attention **within** a single sequence?

“Attention Is All You Need”

Vaswani et al., 2017

Thank You!

Questions?

Priyansh Saxena

All lecture materials available at:

www.priyanshsaxena.com/sst-deep-learning-nn