Why Do We Need Attention Models?

Traditional sequence-to-sequence (Seq2Seq) models, such as those built on **bi-directional RNNs (Recurrent Neural Networks)** or **LSTMs (Long Short-Term Memory networks)**, face limitations in handling long sequences.

- In **standard Seq2Seq**, the encoder processes an input sequence (e.g., a sentence) and compresses all the information into a **single fixed-length vector** (the final hidden state).
- The decoder then generates the output sequence (e.g., translated sentence) based only on this single vector.

The Problem

- When sequences are **long**, a single vector cannot capture all the information.
- As a result, important details from earlier words in the input may be lost.
- Performance degrades significantly for longer sentences, as shown in BLEU score vs. sentence length (longer sentences lead to lower translation accuracy without attention).

The Solution: Attention Mechanism

The **Attention mechanism** was introduced to allow the decoder to "look back" at **all encoder hidden states** (not just the last one). Instead of depending solely on one context vector, the model dynamically **selects relevant parts of the input** for each decoding step.

In essence:
👉 Attention acts like a "searchlight," focusing on the most relevant words in the input sequence while generating each word in the output.

Core Concepts of Attention

1. Encoder Representations

- The encoder (often a **bi-directional RNN/LSTM**) processes the input sequence.
- For each input word, it produces a hidden state h_j.
- So, for an input sequence of length T_x, we have hidden states:

$$h_1, h_2, ..., h_{T_x}$$

These hidden states contain contextual information about each input word.

2. Context Vector (C_i)

For each decoding step iii, the decoder needs a **context vector** C_i that represents the most relevant parts of the input sequence for predicting the next word.

$$C_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

- h_j: Encoder hidden states
- α_{ij}: Attention weights (importance of h_j when predicting output step i)
- T_x: Input sentence length

Thus, C_i is a **weighted sum** of encoder states.

4. Alignment Score (e_{ij})

The alignment score measures how well the inputs around position j match the output at step i.

$$e_{ij} = a(s_{i-1}, h_j)$$

- s_{i-1}: Previous hidden state of the decoder
- h_j: Encoder hidden state at position j
- $a(\cdot)$: A small neural network (often a feedforward layer) that learns to compute compatibility

This function is trainable and learns how to align input tokens with output tokens.

5. Decoder with Attention

At each output step:

1. The decoder computes the alignment scores e_{ij} for all encoder states.
2. Converts them into normalized attention weights α_{ij}.
3. Produces a **context vector C_i** as the weighted sum of encoder states.
4. Combines C_i with the decoder's state to predict the next word.

This allows the decoder to **dynamically focus** on different parts of the input sequence at each step.

Impact on Performance (BLEU Scores)

- BLEU (Bilingual Evaluation Understudy) is a standard metric for evaluating machine translation quality.
- In experiments, Seq2Seq models without attention perform poorly on **longer sentences** because of information loss.
- Models with attention maintain **higher BLEU scores** across longer sentence lengths.

For example:

- Models trained with sentence length limits of 30 vs. 50 show that attention-equipped models (RNNSearch) outperform plain encoder-decoder models (RNNenc).
- The **gap widens as sentence length increases**, proving attention helps capture long-range dependencies.

Key Insights

1. **Why Attention?**
    - Overcomes the bottleneck of a fixed-length context vector.
    - Enables better handling of long sentences and complex dependencies.
2. **What Attention Does?**
    - Computes a **weighted combination** of all encoder states for each decoding step.
    - Learns to focus on the most relevant parts of the input dynamically.
3. **How It Improves Performance?**
    - Provides the decoder with richer, context-dependent information.
    - Maintains translation quality even for long input sequences.
    - Leads to higher BLEU scores compared to vanilla Seq2Seq models.