Lecture 32. NMT

Neural Machine Translation (NMT) is a deep learning approach to translating text from one language to another, offering significant improvements over traditional methods like rule-based or statistical machine translation. Unlike earlier systems, NMT models learn to map source sentences to target sentences end-to-end, capturing complex linguistic patterns and contextual nuances. The **encoder-decoder architecture**, commonly implemented using the **Transformer model**, is the cornerstone of modern NMT systems. This architecture leverages **self-attention** and **cross-attention** mechanisms to process and align source and target languages effectively. Below, I'll explain the encoder-decoder framework, detail the roles of self-attention and cross-attention, and illustrate the process using an English-to-Hindi translation example, ensuring a clear and comprehensive explanation tailored to your request.

---

1. Overview of NMT and Encoder-Decoder Architecture

NMT uses neural networks to translate text by modeling the probability of a target sentence given a source sentence, i.e., $P(\text{target} \mid \text{source})$ P(\text{target} | \text{source}) $P(\text{target} \mid \text{source})$. The **encoder-decoder architecture** is designed to handle this sequence-to-sequence task:

- **Encoder**: Processes the source sentence (e.g., English) to create a compact, context-aware representation of its meaning.
- **Decoder**: Generates the target sentence (e.g., Hindi) token by token, using the encoder's representation and previously generated tokens.

The Transformer model, introduced in the paper *"Attention is All You Need"* (Vaswani et al., 2017), is the most widely used encoder-decoder architecture in NMT. It consists of stacked layers of interconnected nodes, replacing earlier recurrent neural networks (RNNs) with attention mechanisms for better parallelization and performance.

2. Example Sentence for Illustration

Let’s use the following English-to-Hindi translation to ground the explanation:

- **Source Sentence (English)**: "The cat sleeps."
- **Target Sentence (Hindi)**: "बिल्ली सोती है।" (Transliteration: "Billi soti hai.")

We’ll trace how the Transformer’s encoder-decoder processes this sentence, focusing on self-attention and cross-attention.

---

3. Encoder: Processing the Source Sentence

The encoder transforms the source sentence into a set of contextualized representations. Here’s how it works for "The cat sleeps."

Step 1: Tokenization and Embedding

- The sentence is tokenized into individual words or subwords. For simplicity, assume word-level tokenization: ["The", "cat", "sleeps"].
- Each token is converted into a dense vector (embedding) using a pre-trained embedding layer. For example:
  - "The" → $e_{\text{the}} = [0.1, 0.2, ..., 0.5]$
  - "cat" → $e_{\text{cat}} = [0.3, 0.4, ..., 0.7]$
  - "sleeps" → $e_{\text{sleeps}} = [0.5, 0.1, ..., 0.3]$
- **Positional Encoding**: Since Transformers lack sequential processing (unlike RNNs), positional encodings are added to the embeddings to indicate word order. For token i, a positional vector PE_i is added:

$$x_i = e_i + PE_i$$

  So, the input to the encoder is a matrix of vectors: $[x_{\text{the}}, x_{\text{cat}}, x_{\text{sleeps}}]$.

Step 2: Self-Attention in the Encoder

Self-attention allows each token to attend to all other tokens in the source sentence, capturing contextual relationships. For "The cat sleeps," self-attention helps the model understand how "cat" relates to "sleeps" (subject-verb) or "The" (determiner-noun).

- **Mechanism**:
  - For each token, compute **Query (Q)**, **Key (K)**, and **Value (V)** vectors using linear transformations:

    $$Q_i = W_Q x_i, \quad K_i = W_K x_i, \quad V_i = W_V x_i$$

    where W_Q, W_K, W_V are learned weight matrices.
  - Calculate attention scores by comparing each token's query with all keys:

    $$\text{Score}(i, j) = Q_i \cdot K_j$$
  - Normalize scores using softmax to get attention weights:

    $$\alpha_{i,j} = \text{softmax}\left(\frac{Q_i \cdot K_j}{\sqrt{d_k}}\right)$$

    where d_k is the dimension of the key vectors (for scaling).
  - Compute the output for each token as a weighted sum of values:

    $$\text{Attention}_i = \sum_j \alpha_{i,j} V_j$$
- **Multi-Head Attention**: Multiple attention heads run in parallel, each focusing on different aspects (e.g., syntactic vs. semantic relationships), and their outputs are concatenated.
- **Example**: For "cat," self-attention might assign high weights to "sleeps" (indicating the action) and moderate weights to "The" (indicating the determiner). The output vector for "cat" becomes a blend of its own embedding and contextual information from "sleeps" and "The."

4. Decoder: Generating the Target Sentence

The decoder generates the target sentence "बिल्ली सोती है।" (Billi soti hai.) token by token in an autoregressive manner, using the encoder's output and previously generated tokens. Let's break it down.

Step 1: Tokenization and Embedding

- The target sentence is tokenized: ["बिल्ली", "सोती", "है"].
- During training, the decoder receives the full target sequence, shifted right with a start token: ["<start>", "बिल्ली", "सोती", "है"]. During inference, it generates one token at a time.</start>
- Each token is embedded and augmented with positional encodings, similar to the encoder. For example:

The decoder uses **masked self-attention** to ensure that each target token only attends to previous tokens, preserving the autoregressive property (future tokens are not seen during prediction).

- **Mechanism**:
  - Similar to encoder self-attention, but a mask prevents attention to future positions. For "बिल्ली," the model attends only to "<start>" and itself, not "सोती" or "है."</start>
  - Compute Q, K, V vectors for each target token.
  - Apply masking to the attention scores (set future scores to $-\infty$ before softmax).
  - Output a context vector for each token, reflecting its relationship with prior tokens.
- **Example**: When generating "सोती," masked self-attention allows the model to focus on "<start>" and "बिल्ली," capturing that "बिल्ली" is the subject performing an action.</start>

Step 3: Cross-Attention (Encoder-Decoder Attention)

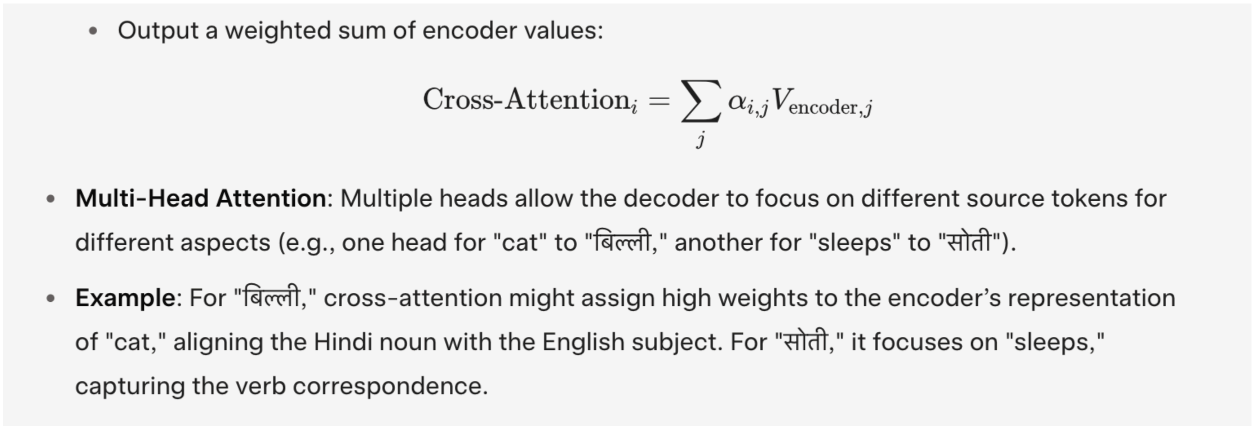
Cross-attention aligns the target tokens with the source sentence's representations from the encoder. This is where the decoder "looks" at the encoded "The cat sleeps" to generate "बिल्ली सोती है।"

- **Mechanism**:
  - The decoder's self-attention output provides query vectors Q_{decoder}.
  - The encoder's output provides key and value vectors: $K_{\text{encoder}}, V_{\text{encoder}}$.
  - Compute attention scores:

$$\text{Score}(i, j) = Q_{\text{decoder},i} \cdot K_{\text{encoder},j}$$

  - Normalize with softmax to get weights:

$$\alpha_{i,j} = \text{softmax}\left(\frac{Q_{\text{decoder},i} \cdot K_{\text{encoder},j}}{\sqrt{d_k}}\right)$$



- Output a weighted sum of encoder values:

$$\text{Cross-Attention}_i = \sum_j \alpha_{i,j} V_{\text{encoder},j}$$

- **Multi-Head Attention**: Multiple heads allow the decoder to focus on different source tokens for different aspects (e.g., one head for "cat" to "बिल्ली," another for "sleeps" to "सोती").
- **Example**: For "बिल्ली," cross-attention might assign high weights to the encoder's representation of "cat," aligning the Hindi noun with the English subject. For "सोती," it focuses on "sleeps," capturing the verb correspondence.

Advantages

1. **End-to-End Learning**: NMT trains a single neural network to map source to target sentences directly, eliminating the need for handcrafted rules (RBMT) or separate models (SMT). For example, in translating "The cat sleeps" to "बिल्ली सोती है।," NMT learns to handle Hindi's SOV order and gender agreement without explicit linguistic rules.
2. **Contextual Understanding via Self-Attention**: The Transformer's self-attention mechanism captures long-range dependencies and contextual relationships within the source and target sentences, unlike RBMT's local rules or SMT's fixed-phrase alignments. This ensures accurate subject-verb agreement, such as "बिल्ली" (feminine) with "सोती" in Hindi.
3. **Dynamic Alignment with Cross-Attention**: Cross-attention aligns target tokens with relevant source tokens, handling linguistic differences like word order (English SVO vs. Hindi SOV) and morphological nuances (e.g., adding "है" for present tense). This outperforms SMT's rigid statistical alignments, which may miss structural nuances.
4. **Robust Morphological Handling**: NMT learns complex morphological patterns, such as Hindi's gender and number agreement, directly from data, reducing errors compared to RBMT's labor-intensive rule creation. For instance, NMT correctly inflects "सोती" for the feminine "बिल्ली" without manual rules.
5. **Improved Fluency and Generalization**: NMT produces more fluent and natural translations by modeling entire sentences, unlike SMT's phrase-based approach, which can lead to disjointed outputs. For "The cat sleeps," NMT ensures a cohesive "बिल्ली सोती है।," capturing Hindi's syntactic flow.
6. **Scalability and Efficiency**: Transformers use parallel processing via attention mechanisms, making NMT faster and more scalable than RNN-based SMT or rule-heavy RBMT. This allows efficient handling of large datasets and complex sentences across languages like English and Hindi.