Transformer Architecture in NLP: Day 1 - Introduction to Sequence-to-Sequence Models

Objective: Today, we will lay the groundwork for understanding the Transformer architecture by exploring the fundamental concepts of sequence-to-sequence (Seq2Seq) models. We'll discuss why they are necessary and the limitations of earlier approaches.

1. What are Sequence-to-Sequence Models?

At its core, a sequence-to-sequence model is a type of neural network designed to process an input sequence and generate an output sequence. These sequences don't have to be of the same length. Think of it like translation: you take an input sentence in one language (a sequence of words) and produce an output sentence in another language (another sequence of words), which might have a different number of words.

Examples of Seq2Seq Tasks:

* **Machine Translation:** English to French, Spanish to German.
* **Text Summarization:** Condensing a long article into a shorter summary.
* **Question Answering:** Taking a question and a passage of text, and generating an answer.
* **Chatbots:** Generating a response to a user's input.

2. The Need for Seq2Seq Models

Before Seq2Seq models became prominent, tasks involving variable-length input and output sequences were challenging. Traditional feed-forward neural networks expect fixed-size inputs and produce fixed-size outputs, making them unsuitable for tasks like translation where sentence lengths

vary greatly.

3. Early Approaches and Their Limitations

Early attempts often involved complex, multi-step processes or simpler models that struggled with long sequences. The primary limitations included:

* **Fixed-Length Context Vector:** Early encoder-decoder models (a precursor to Transformers) compressed the entire input sequence into a single, fixed-length "context vector." This became a bottleneck for long sentences, as it was difficult to capture all the nuances and information in one vector.
* **Information Loss:** Compressing long sequences led to a loss of information, especially for words appearing later in the input.
* **Vanishing Gradients:** Recurrent Neural Networks (RNNs), often used in these early models, suffered from the vanishing gradient problem, making it hard to learn long-range dependencies (relationships between words far apart in a sequence).

4. Introducing the Encoder-Decoder Framework

The encoder-decoder architecture was a significant step forward. It consists of two main parts:

* **Encoder:** Reads the input sequence and compresses it into a context vector (or a series of hidden states).
* **Decoder:** Takes the context vector and generates the output sequence, one element at a time.

While an improvement, the fixed-length context vector limitation persisted. This is where the Transformer architecture introduces a revolutionary change, which we will explore tomorrow!

Key Takeaway for Day 1: Seq2Seq models are crucial for tasks involving variable-length input and output sequences. Early encoder-decoder models showed promise but were hampered by information bottlenecks. The Transformer architecture aims to overcome these limitations through a novel approach.

Looking Ahead: Tomorrow, we'll dive into the groundbreaking mechanism that powers the Transformer: self-attention!