Explanation of the Self-Attention Mechanism in the Transformer Architecture

Claude (Anthropic), v.3.5 Sonnet
Prompted by Murat Yıldızoğlu

February, 10, 2025

The **self-attention mechanism** is a core component of the Transformer architecture, introduced in the seminal paper *"Attention is All You Need"* by Vaswani et al. (2017). It allows the model to focus on relevant parts of the input sequence when processing each token, enabling it to capture relationships between words regardless of their distance in the sequence. This mechanism is crucial for handling long-range dependencies in text, making Transformers highly effective for tasks like machine translation, text summarization, and question answering.

1 How Self-Attention Works

The self-attention mechanism computes a weighted representation of the input sequence for each token by considering its relationship with all other tokens in the sequence. Here's a step-by-step explanation:

1.1 Input Representation

Each word in the input sequence is first converted into a vector representation (e.g., word embeddings). For example, consider the sentence:

Input Sentence: "The cat sat on the mat."

Each word is represented as a vector:

$$\text{Input} = [\mathbf{e}_{\text{The}}, \mathbf{e}_{\text{cat}}, \mathbf{e}_{\text{sat}}, \mathbf{e}_{\text{on}}, \mathbf{e}_{\text{the}}, \mathbf{e}_{\text{mat}}]$$

1.2 Query, Key, and Value Matrices

For each word, the model computes three vectors:

- **Query (Q)**: Represents the word we are focusing on.
- **Key (K)**: Represents the word we are comparing against.
- **Value (V)**: Contains the information to be aggregated.

These vectors are computed by multiplying the input embeddings with learned weight matrices:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

Where:

- X is the input matrix (word embeddings).
- W^Q, W^K, and W^V are learned weight matrices.

1.3 Scaled Dot-Product Attention

The attention mechanism computes a score for each pair of words in the sequence to determine how much attention one word should pay to another. This is done using the dot product of the Query and Key vectors, scaled by the square root of the dimension of the Key vectors (d_k):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- **Dot Product (QK^T)**: Measures the similarity between the Query and Key vectors.
- **Scaling ($\sqrt{d_k}$)**: Prevents the dot product values from becoming too large, which could lead to very small gradients.
- **Softmax**: Converts the scores into probabilities, ensuring they sum to 1.

1.4 Weighted Sum of Values

The attention scores are used to compute a weighted sum of the Value vectors. This produces a new representation for each word that incorporates information from the entire sequence.

2 Example of Self-Attention

2.1 Input Sentence

: "The cat sat on the mat."

1. **Step 1: Compute Query, Key, and Value Vectors**

   For simplicity, assume the embeddings for "cat" are:

   - Query (Q_{cat}) = [0.2, 0.1]
   - Key (K_{The}) = [0.3, 0.4]
   - Value (V_{The}) = [0.5, 0.6]

   Similarly, compute Q, K, and V for all other words.

2. **Step 2: Compute Attention Scores**

   For "cat," compute the dot product of Q_{cat} with the Key vectors of all words:

   $$\text{Score}_{\text{cat, The}} = Q_{\text{cat}} \cdot K_{\text{The}} = (0.2 \cdot 0.3) + (0.1 \cdot 0.4) = 0.1$$

   Repeat this for all other words in the sequence.

3. **Step 3: Apply Softmax**

   Convert the scores into probabilities using the softmax function:

   $$\text{Softmax(Scores)} = \frac{\exp(\text{Score})}{\sum \exp(\text{Scores})}$$

4. **Step 4: Compute Weighted Sum**

   Use the attention probabilities to compute a weighted sum of the Value vectors. For example:

   $$\text{Output}_{\text{cat}} = \sum(\text{Attention Probability} \cdot V)$$

   This output vector represents "cat" in the context of the entire sentence.

3 Key Features of Self-Attention

1. **Context-Aware Representations**:
   - Each word's representation is updated to include information from all other words in the sequence.
   - Example: In "The cat sat on the mat," the representation of "cat" will include information about "sat" and "mat."
2. **Long-Range Dependencies**:
   - Self-attention can capture relationships between words that are far apart in the sequence.
   - Example: In "The cat sat on the mat because it was tired," the word "it" can attend to "cat."
3. **Parallelization**:
   - Unlike RNNs, which process sequences sequentially, self-attention processes all words in parallel, making it computationally efficient.

4 Multi-Head Attention

In practice, Transformers use **multi-head attention**, where multiple self-attention mechanisms are applied in parallel. Each head learns to focus on different aspects of the input, such as syntax or semantics. The outputs of all heads are concatenated and linearly transformed to produce the final representation.

5 Applications of Self-Attention

1. **Machine Translation**:
   - Self-attention helps align words in the source and target languages.
   - Example: Translating "The cat sat on the mat" to French.
2. **Text Summarization**:
   - Self-attention identifies the most important parts of a document to generate a summary.
3. **Question Answering**:
   - Self-attention helps the model focus on relevant parts of the context to answer a question.

6 Conclusion

The self-attention mechanism is the foundation of the Transformer architecture, enabling it to process sequences efficiently and capture complex relationships between words. By computing context-aware representations for each word, self-attention has revolutionized NLP tasks, making Transformers the backbone of modern language models like BERT, GPT, and T5.