Introduction to Transformers

Dr. Akhtar Jamil
Department of Computer Science
National University of Computer and Emerging Sciences,
Islamabad, Pakistan

1 Introduction

Let's assume we have the following sentence:
"Akhtar teaches Generative AI"

To use words as input for our model, they must first be transformed into a numeric representation, known as word embeddings. This can be done using pre-trained models like Word2Vec, GloVe, or BERT, which capture the semantic meaning of words and encode them into vectors. Let's treat each word in the sentence as a single token (it is possible to break a word into subword during tokenization). Let us now perform each step to pass the input through the transformer.

1.1 Token Embeddings

Let's assume the following 3-dimensional embeddings for each word:
- "Akhtar": $[0.6, 0.1, 0.8]$
- "teaches": $[0.5, 0.9, 0.7]$
- "Generative": $[0.4, 0.2, 0.9]$
- "AI": $[0.7, 0.3, 0.6]$

So, the input embeddings matrix, after putting all these words, for the sentence is:

$$\text{Input Embeddings} = \begin{bmatrix} 0.6 & 0.1 & 0.8 \\ 0.5 & 0.9 & 0.7 \\ 0.4 & 0.2 & 0.9 \\ 0.7 & 0.3 & 0.6 \end{bmatrix}$$

In this matrix, each row represents a single word.

1.2 Positional Encoding

We will now generate positional encodings for each position of each word in the sentence. The formula for positional encoding is:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}\right)$$

For simplicity, let's calculate positional encodings for each word at position 0, 1, 2, and 3 for a 3-dimensional embedding.

Step 3: Calculate Positional Encodings

- For position 0 (Akhtar):

$$PE(0) = [\sin(0), \cos(0), \sin(0)] = [0, 1, 0]$$

- For position 1 (teaches):

$$PE(1) = [\sin\left(\frac{1}{10000^{\frac{0}{3}}}\right), \cos\left(\frac{1}{10000^{\frac{1}{3}}}\right), \sin\left(\frac{1}{10000^{\frac{2}{3}}}\right)] = [\sin(1), \cos(1), \sin(1{\times}10^{-0.666})]$$

Using approximate values:

$$PE(1) = [0.8415, 0.5403, 1 \times 10^{-4}]$$

- For position 2 (Generative):

$$PE(2) = [\sin\left(\frac{2}{10000^{\frac{0}{3}}}\right), \cos\left(\frac{2}{10000^{\frac{1}{3}}}\right), \sin\left(\frac{2}{10000^{\frac{2}{3}}}\right)]$$

Approximate values:

$$PE(2) = [0.9093, -0.4161, 2 \times 10^{-4}]$$

- For position 3 (AI):

$$PE(3) = [\sin(3), \cos(3), \sin(3 \times 10^{-0.6667})]$$

Approximate values:

$$PE(3) = [0.1411, -0.9899, 3 \times 10^{-4}]$$

Add Positional Encodings to Input Embeddings Now we add the positional encodings to the input embeddings to form the Final Embeddings (X) that will be fed to the transformer.

$$\text{X} = \text{Input Embeddings} + \text{Positional Encodings}$$

$$X = \begin{bmatrix} 0.6+0 & 0.1+1 & 0.8+0 \\ 0.5+0.8415 & 0.9+0.5403 & 0.7+1\times10^{-4} \\ 0.4+0.9093 & 0.2+(-0.4161) & 0.9+2\times10^{-4} \\ 0.7+0.1411 & 0.3+(-0.9899) & 0.6+3\times10^{-4} \end{bmatrix}$$

Which results in:

$$X = \begin{bmatrix} 0.6 & 1.1 & 0.8 \\ 1.3415 & 1.4403 & 0.7001 \\ 1.3093 & -0.2161 & 0.9002 \\ 0.8411 & -0.6899 & 0.6003 \end{bmatrix}$$

1.3 Scaled Dot-Product Attention

First, we need to obtain query (Q), key (K) and values (V) matrices. These can be calculated by learning the transformation with some weight Ws. Let us assume that the following weight matrices are used for this example. In practice these three matrices will be learnt during training.

Weight matrices:

$$W_Q = W_K = W_V = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix}$$

The formula for Scaled Dot Product Attention is:

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

Where:

1. Q : Query matrix (represents the queries from the input sequence).
2. K : Key matrix (represents the keys from the input sequence).
3. V : Value matrix (represents the values from the input sequence).
4. $QK^\top$: Dot product of the Query and Key matrices.
5. $\sqrt{d_k}$: Scaling factor, where d_k is the dimension of the Key vector.
   This helps prevent extremely large dot-product values.
6. Softmax: A function applied to normalize the attention scores (row-wise) into probabilities.
7. V : Weighted values are computed using the normalized attention scores.

1.3.1 Compute Q, K, and V:

To apply scaled dot product attention, we need to compute Q, K, and V.

$$Q = X \cdot W_Q$$

$$K = \mathrm{X} \cdot W_K$$
$$V = \mathrm{X} \cdot W_V$$

Let us first compute Q. Multiply each row of the X with W_Q:

$$Q_1 = [0.6, 1.1, 0.8] \cdot \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} = [1.25, 1.25, 1.25]$$

$$Q_2 = [1.3415, 1.4403, 0.7001] \cdot \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} = [1.74095, 1.74095, 1.74095]$$

$$Q_3 = [1.3093, -0.2161, 0.9002] \cdot \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} = [0.9967, 0.9967, 0.9967]$$

$$Q_4 = [0.8411, -0.6899, 0.6003] \cdot \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{bmatrix} = [0.37575, 0.37575, 0.37575]$$

Thus:

$$Q = \begin{bmatrix} 1.25 & 1.25 & 1.25 \\ 1.74095 & 1.74095 & 1.74095 \\ 0.9967 & 0.9967 & 0.9967 \\ 0.37575 & 0.37575 & 0.37575 \end{bmatrix}$$

Similarly, $K = Q$ and $V = Q$, as the weight matrices are the same.

1.3.2 Compute QK^T

To compute QK^T, multiply Q with the transpose of K:

$$K^T = \begin{bmatrix} 1.25 & 1.74095 & 0.9967 & 0.37575 \\ 1.25 & 1.74095 & 0.9967 & 0.37575 \\ 1.25 & 1.74095 & 0.9967 & 0.37575 \end{bmatrix}$$

Now compute:

$$\mathrm{QK}^T = Q \cdot K^T$$

For row 1:

$$[1.25, 1.25, 1.25] \cdot \begin{bmatrix} 1.25 \\ 1.25 \\ 1.25 \end{bmatrix} = 1.25 \cdot 1.25 + 1.25 \cdot 1.25 + 1.25 \cdot 1.25 = 4.6875$$

Similarly, compute for all rows:

$$\mathrm{QK}^T = \begin{bmatrix} 4.6875 & 6.5286 & 3.7376 & 1.4091 \\ 6.5286 & 9.0927 & 5.2056 & 1.9625 \\ 3.7376 & 5.2056 & 2.9802 & 1.1235 \\ 1.4091 & 1.9625 & 1.1235 & 0.4236 \end{bmatrix}$$

1.3.3 Scale QK^T by $\sqrt{d_k}$

$$\text{Scale Factor} = \sqrt{d_k} = \sqrt{3} \approx 1.732$$

Divide each element in QK^T by 1.732:

$$\text{Scaled } QK^T = \begin{bmatrix} 2.7063 & 3.7693 & 2.1579 & 0.8135 \\ 3.7693 & 5.2497 & 3.0055 & 1.1330 \\ 2.1579 & 3.0055 & 1.7206 & 0.6487 \\ 0.8135 & 1.1330 & 0.6487 & 0.2445 \end{bmatrix}$$

1.3.4 Apply Softmax for Calculate Attention Scores

Apply softmax to each row of Scaled QK^T:

Softmax formula:

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

For the first row:

$$\text{Row 1: } [2.7063, 3.7693, 2.1579, 0.8135]$$

Exponentiate:

$$e^{2.7063} = 14.954,\ e^{3.7693} = 43.338,\ e^{2.1579} = 8.647,\ e^{0.8135} = 2.255$$

Sum: $14.954 + 43.338 + 8.647 + 2.255 = 69.194$

Softmax:

$$\text{Row 1: } [0.216, 0.626, 0.125, 0.033]$$

Repeat for all rows:

$$\text{Attention Scores} = \begin{bmatrix} 0.216 & 0.626 & 0.125 & 0.033 \\ 0.169 & 0.741 & 0.079 & 0.012 \\ 0.238 & 0.556 & 0.154 & 0.053 \\ 0.264 & 0.363 & 0.224 & 0.149 \end{bmatrix}$$

1.3.5 Final Output

$$\text{Output} = \text{Attention Scores} \cdot V$$

For row 1:

$$[0.216, 0.626, 0.125, 0.033] \cdot \begin{bmatrix} 1.25 & 1.25 & 1.25 \\ 1.74095 & 1.74095 & 1.74095 \\ 0.9967 & 0.9967 & 0.9967 \\ 0.37575 & 0.37575 & 0.37575 \end{bmatrix} = [1.497, 1.497, 1.497]$$

Repeat for all rows and get the final attention output:

$$\text{Attention Output} = \begin{bmatrix} 1.497 & 1.497 & 1.497 \\ 1.583 & 1.583 & 1.583 \\ 1.438 & 1.438 & 1.438 \\ 1.241 & 1.241 & 1.241 \end{bmatrix}$$