Study note on transformer

Baohua Zhou
Yale University
baohua.x.zhou@gmail.com

Abstract

This is an informal study note about the original transformer paper [1], and the goal is to lay out the mathematical details for a sequence-to-sequence task.

Preliminary

Here, we try to lay out the whole mathematical structure of a transformer model on an example of language translation [1, 2]. Suppose that we want to translate a sentence with n words in one language $[x_1', x_2', \ldots, x_i', \ldots, x_n']$ into a sentence with m words in another language $[y_1', y_2', \ldots, y_c', \ldots, y_m']$, where $x_i' \in V_x$ and $y_c' \in V_y$. V_x and V_y are the collections of all words in the two languages, respectively, and they have N_x and N_y words, respectively. We assume that all the words are represented as one-hot vectors

$$x_i' = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \qquad y_c' = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

where the length of the vector x_i' is N_x, and the length of the vector y_c' is N_y. All words are embedded into d-vectors with some learned word embedding [1]:

$$x_i = \text{SomeLearnedWordEmbedding}(x_i') \in \mathbb{R}^d$$

$$y_c = \text{SomeLearnedWordEmbedding}(y_c') \in \mathbb{R}^d$$

Positional encoding might be needed since the word order in a sentence matters, although I don't think it is necessary at all [1]:

$$p_i \in \mathbb{R}^d, (p_i)_{2l} = \sin\left(\frac{i}{10000^{2l/d}}\right), (p_i)_{2l+1} = \cos\left(\frac{i}{10000^{2l/d}}\right), l = 0, 1, \ldots, d/2$$

We combine the word embedding and the positional encoding by adding them together [1]:

$$x_i \leftarrow x_i + p_i \in \mathbb{R}^d$$

$$y_c \leftarrow y_c + p_c \in \mathbb{R}^d$$

You may rescale the word embedding by a factor of $\sqrt{d}$ in certain cases.

Encoder

Now, we construct the encoder. An encoder contains N identical blocks of computing units, and each computing unit is a mapping $f_\theta: \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$, where θ represents the trainable parameters. For the first block, the input is $X = [x_1, x_2, \dots, x_i, \dots, x_n] \in \mathbb{R}^{d\times n}$ and the output is $Z = [z_1, z_2, \dots, z_i, \dots, z_n] \in \mathbb{R}^{d\times n}$. Let's break down the mapping f_θ. Each column of the input is mapped to three types of vectors, called the queries $Q^{(h)}(x_i) \in \mathbb{R}^k$, the keys $K^{(h)}(x_i) \in \mathbb{R}^k$, and the values $V^{(h)}(x_i) \in \mathbb{R}^k$, where $h = 1, 2, \dots, H$ indicates different functions in different attention heads. The simplest forms of the $3H$ functions, Q, K, V, could be linear transformations by $3H$ matrices, respectively [2]. The core concept of the transformer is the acausal self-attention weights defined as some normalized kernels:

$$\alpha_{ij}^{(h)} = \frac{\exp\left[\frac{\langle Q^{(h)}(x_i), K^{(h)}(x_j)\rangle}{\sqrt{k}}\right]}{\sum_{j'=1}^{n} \exp\left[\frac{\langle Q^{(h)}(x_i), K^{(h)}(x_{j'})\rangle}{\sqrt{k}}\right]} \in \mathbb{R}$$

where $\langle\cdot,\cdot\rangle$ represents the dot product. The self-attention value for x_i is [2]:

$$u_i' = \sum_{h=1}^{H} W_O^{(h)} \sum_{j=1}^{n} \alpha_{ij}^{(h)} V^{(h)}(x_j) \in \mathbb{R}^d$$

where $W_O^{(h)} \in \mathbb{R}^{d\times k}$. A residual connection is used [3], followed by a layer normalization [4]:

$$u_i = \mathrm{LayerNorm}(u_i' + x_i; \gamma_1, \beta_1) \in \mathbb{R}^d$$

A feedforward layer follows with the residual connection and the layer normalization:

$$z_i' = W_2 \mathrm{ReLU}(W_1 u_i + b_1) \in \mathbb{R}^d$$

$$z_i = \mathrm{LayerNorm}(z_i' + u_i; \gamma_2, \beta_2) \in \mathbb{R}^d$$

where $W_1 \in \mathbb{R}^{d_f\times d}$, $W_2 \in \mathbb{R}^{d\times d_f}$, $b_1 \in \mathbb{R}^{d_f}$.

The LayerNorm function on a d-vector v is defined as [5]:

$$\mathrm{LayerNorm}(v; \gamma, \beta) = \gamma \odot \frac{v - \mu}{\sigma} + \beta \qquad \gamma, \beta \in \mathbb{R}^d$$

$$\mu = \frac{1}{d}\sum_{l=1}^{d} v_l,\ \sigma = \sqrt{\frac{1}{d}\sum_{l=1}^{d}(v_l - \mu)^2}$$

where ⨀ represents element-wise multiplications.

Thus, we have defined the mapping for the first block of the computing unit $z_i = f_\theta(x_i)$, and the trainable parameters θ include the parameters in the functions $Q^{(h)}, K^{(h)}, V^{(h)}$, and matrices $W_O^{(h)}$, W_1, W_2, and vectors γ_1, β_1, γ_2, β_2, b_1. The whole encoder applies this mapping N times:

$$z_i^{(N)} = f_{\theta_N} \circ \cdots \circ f_{\theta_1}(x_i) \in \mathbb{R}^d, \qquad i = 1, 2, \ldots, n$$

Note that all trainable parameters are independently trained in blocks 1 to N.

Decoder

Like the encoder, the decoder also constitutes N identical blocks of computing units followed by a linear transform and a softmax operation. To be specific, we assume that we have already had the first c words in the translated sentence: $y_1, y_2, \ldots, y_c$, and the decoder is trying to predict the $(c+1)$th word in the target language. With this assumption, each block is then a mapping $g_{\theta'}: \mathbb{R}^{d\times c} \to \mathbb{R}^{d\times c}$, and has three sub-layers. The first layer is the multi-head causal self-attention layer with the queries $Q'^{(h)}(y_j) \in \mathbb{R}^k$, the keys $K'^{(h)}(y_j) \in \mathbb{R}^k$, and the values $V'^{(h)}(y_j) \in \mathbb{R}^k$, where $j = 1, 2, \ldots, c$ and $h = 1, 2, \ldots, H$ indicates different functions in different attention heads. We can calculate the causal attention weights:

$$\alpha'^{(h)}_{jr} = \frac{\exp\left[\frac{\langle Q'^{(h)}(y_j), K'^{(h)}(y_r)\rangle}{\sqrt{k}}\right]}{\sum_{r'=1}^{j} \exp\left[\frac{\langle Q'^{(h)}(y_j), K'^{(h)}(y_{r'})\rangle}{\sqrt{k}}\right]}$$

The attention value for y_j is:

$$v'_j = \sum_{h=1}^{H} W'^{(h)}_O \sum_{r=1}^{j} \alpha'^{(h)}_{jr} V'^{(h)}(y_r) \in \mathbb{R}^d$$

where $W'^{(h)}_O \in \mathbb{R}^{d\times k}$. Note that the second sum in the above equation is only up to j, which is the reason that this is a causal attention layer compared with the acausal attention layer in the encoder. A residual connection is used, followed by a layer normalization:

$$v_j = \text{LayerNorm}\left(v'_j + y_j; \gamma_3, \beta_3\right) \in \mathbb{R}^d$$

A decoder block has a second attention layer that has the keys $K''^{(h)}\left(z_i^{(N)}\right) \in \mathbb{R}^k$ and the values $V''^{(h)}\left(z_i^{(N)}\right) \in \mathbb{R}^k$ from the final output of the encoder, and the queries from the previous attention layer of the decoder block, $Q''^{(h)}(v_j) \in \mathbb{R}^k$. We can calculate the attention weights for this layer:

$$\alpha''^{(h)}_{ji} = \frac{\exp\left[\frac{\langle Q''^{(h)}(v_j), K''^{(h)}(z_i^{(N)})\rangle}{\sqrt{k}}\right]}{\sum_{i'=1}^{n} \exp\left[\frac{\langle Q''^{(h)}(v_j), K''^{(h)}(z_{i'}^{(N)})\rangle}{\sqrt{k}}\right]}$$

The attention value for v_j is:

$$w_j' = \sum_{h=1}^{H} W_O''^{(h)} \sum_{i=1}^{n} \alpha''^{(h)}_{ji} V''^{(h)}(z_i^{(N)}) \in \mathbb{R}^d$$

where $W_O''^{(h)} \in \mathbb{R}^{d \times k}$. A residual connection is used, followed by a layer normalization:

$$w_j = \text{LayerNorm}(w_j' + v_j; \gamma_4, \beta_4) \in \mathbb{R}^d$$

A feedforward layer follows with the residual connection and the layer normalization:

$$o_j' = W_4 \text{ReLU}(W_3 w_j + b_2) \in \mathbb{R}^d$$

$$o_j = \text{LayerNorm}(o_j' + w_j; \gamma_5, \beta_5) \in \mathbb{R}^d$$

where $W_3 \in \mathbb{R}^{d_f \times d}$, $W_4 \in \mathbb{R}^{d \times d_f}$and $b_2 \in \mathbb{R}^{d_f}$.

We have thus defined the mapping for the first block of the computing unit $o_j = g_{\theta'}(y_j)$, and the trainable parameters θ' include the parameters in the functions $Q''^{(h)}, K''^{(h)}, V''^{(h)}$, and matrices $W_O'^{(h)}, W_O''^{(h)}$, W_3, W_4, and vectors γ_3, β_3, γ_4, β_4, γ_5, β_5, b_3. The decoder applies this mapping N times:

$$o_j^{(N)} = g_{\theta_N'} \circ \cdots \circ g_{\theta_1'}(y_j) \in \mathbb{R}^d, \qquad j = 1, 2, \ldots, c$$

Note that all trainable parameters are independently trained in blocks 1 to N.

A linear operation $W_F \in \mathbb{R}^{N_y \times d}$ transforms $o_c^{(N)}$ to a N_y-vector that is sent to a softmax function to get probabilities over the entire vocabulary of the target language:

$$p = \text{softmax}\left[W_F o_c^{(N)}\right] \in \mathbb{R}^{N_y}$$

The word with the highest probability is assigned as the $(c+1)$th word in the target sentence.

Acknowledgement

The author thanks Awni Altabaa and StatQuest with Josh Starmer for clarifying certain concepts.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. & Kaiser, L. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

[2] Thickstun, J. (2021). The transformer model in equations. *University of Washington: Seattle, WA, USA*.

[3] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[4] Ba, J. L., Kiros, J. R. & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.