Mathematical Framework for Transformer-Based Machine Translation

Ankit Pratap Singh

1 Problem Setup

We have a dataset of N parallel sentence pairs:

$$\{(X^{(i)}, Y^{(i)})\}_{i=1}^{N}$$

where:

- $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_{n_i}^{(i)})$ is the **source sentence** of length n_i.
- $Y^{(i)} = (y_1^{(i)}, y_2^{(i)}, ..., y_{m_i}^{(i)})$ is the **target sentence** of length m_i.

2 Encoder: Processing the Source Sentence

2.1 Token Embedding

Each word $x_j^{(i)}$ is represented as a one-hot vector $x_j^{(i)} \in \mathbb{R}^{V_s}$, where V_s is the source vocabulary size.

The **embedding matrix** W_E^X is of shape:

$$W_E^X \in \mathbb{R}^{V_s \times d}$$

The word embedding $h_j^{(i)} \in \mathbb{R}^d$ is obtained by:

$$h_j^{(i)\top} = x_j^{(i)\top} W_E^X$$

For the entire sentence:

$$H^{(i)} = X^{(i)\top} W_E^X$$

where:

- $X^{(i)} \in \mathbb{R}^{V_s \times n_i}$ is the one-hot representation.
- $H^{(i)} \in \mathbb{R}^{n_i \times d}$ contains word embeddings.

2.2 Add Positional Encoding

Since the Transformer lacks recurrence, we **add positional encodings**:

$$\tilde{H}^{(i)} = H^{(i)} + P$$

where $P \in \mathbb{R}^{n_i \times d}$ is a fixed positional encoding matrix.

2.3 Self-Attention in Encoder

Compute **queries, keys, and values**:

$$Q^{(i)} = \tilde{H}^{(i)} W_Q^E, \quad K^{(i)} = \tilde{H}^{(i)} W_K^E, \quad V^{(i)} = \tilde{H}^{(i)} W_V^E$$

where:

- $W_Q^E, W_K^E, W_V^E \in \mathbb{R}^{d \times d}$.
- $Q^{(i)}, K^{(i)}, V^{(i)} \in \mathbb{R}^{n_i \times d}$.

Compute **scaled dot-product attention**:

$$A^{(i)} = \text{softmax}\left(\frac{Q^{(i)} K^{(i)T}}{\sqrt{d}}\right) V^{(i)}$$

Apply a **feedforward network**:

$$H_E^{(i)} = \text{ReLU}(A^{(i)} W_1^E) W_2^E$$

where:

- $W_1^E, W_2^E \in \mathbb{R}^{d \times d}$.
- $H_E^{(i)} \in \mathbb{R}^{n_i \times d}$.

3 Decoder: Processing the Target Sentence

3.1 Token Embedding

Each target word $y_j^{(i)}$ is represented as a one-hot vector $y_j^{(i)} \in \mathbb{R}^{V_t}$, where V_t is the target vocabulary size.

The **target embedding matrix** is:

$$W_E^Y \in \mathbb{R}^{V_t \times d}$$

The word embedding $g_j^{(i)} \in \mathbb{R}^d$ is obtained as:

$$g_j^{(i)\top} = y_j^{(i)\top} W_E^Y$$

For the entire sequence:

$$G^{(i)} = Y^{(i)\top} W_E^Y$$

where:

- $Y^{(i)} \in \mathbb{R}^{V_t \times m_i}$.
- $G^{(i)} \in \mathbb{R}^{m_i \times d}$.

3.2 Masked Self-Attention

To prevent the decoder from attending to future words during training, we apply a **causal masking matrix** M, defined as:

$$M_{j,k} = \begin{cases} 0, & \text{if } k \leq j \text{ (word can attend to itself and past words)} \\ -\infty, & \text{if } k > j \text{ (future words are masked)} \end{cases}$$

For example, if the decoder processes a sentence of length $m = 4$, the mask matrix M is:

$$M = \begin{bmatrix} 0 & -\infty & -\infty & -\infty \\ 0 & 0 & -\infty & -\infty \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

When added to the attention logits before applying the softmax function, this ensures that the decoder cannot see future tokens.

Compute:

$$Q_D^{(i)} = G^{(i)} W_Q^D, \quad K_D^{(i)} = G^{(i)} W_K^D, \quad V_D^{(i)} = G^{(i)} W_V^D$$

Apply the **masked attention**:

$$A_D^{(i)} = \text{softmax}\left(\frac{Q_D^{(i)} K_D^{(i)T} + M}{\sqrt{d}} \right) V_D^{(i)}$$

3.3 Cross-Attention With Encoder Output

$$Q_C^{(i)} = A_D^{(i)} W_Q^C, \quad K_C^{(i)} = H_E^{(i)} W_K^C, \quad V_C^{(i)} = H_E^{(i)} W_V^C$$

$$A_C^{(i)} = \text{softmax}\left(\frac{Q_C^{(i)} K_C^{(i)T}}{\sqrt{d}} \right) V_C^{(i)}$$

3.4 Final Feedforward and Softmax

$$H_D^{(i)} = \text{ReLU}(A_C^{(i)} W_1^D) W_2^D$$

$$\hat{Y}^{(i)\top} = \text{softmax}(H_D^{(i)} W_O)$$

where $W_O \in \mathbb{R}^{d \times V_t}$ projects to the vocabulary.
$\hat{Y}^{(i)\top} = [\hat{y}_1^{(i)}, ..., \hat{y}_{m_i}^{(i)}]^\top \in \mathbb{R}^{m_i \times V_t}$

4 Loss Function and Optimization

The **cross-entropy loss** is:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{m_i} \sum_{k=1}^{V_t} y_j^{(i)}[k] \log \hat{y}_j^{(i)}[k]$$

where

- $y_j^{(i)}[k]$ is **1 if the correct word is** k, otherwise 0.
- $\hat{y}_j^{(i)}[k]$ is the predicted conditional probability for word k given it has seen $y_{j'<j}^{(i)}$ and $X^{(i)}$.

We optimize by updating weights via **gradient descent**:

$$W \leftarrow W - \eta \frac{\partial \mathcal{L}}{\partial W}$$

4.1 Representing loss in terms of weights recursively

Since

$$\hat{Y}^{(i)\top} = \text{softmax}(H_D^{(i)} W_O)$$

Thus, for any word k at position j in sentence i:

$$\hat{y}_j^{(i)}[k] = \text{softmax}(H_D^{(i)} W_O)_{j,k}$$

And

$$H_D^{(i)} = \text{ReLU}(A_C^{(i)} W_1^D) W_2^D$$

Implies

$$\hat{y}_j^{(i)}[k] = \text{softmax}(\text{ReLU}(A_C^{(i)} W_1^D) W_2^D W_O)_{j,k}$$

In this way recursively substituting for values we can express loss in terms of weights to be optimized.

5 Multi-Head Attention Mechanism

Instead of using a single self-attention mechanism, we can apply **Multi-Head Attention (MHA)**, where multiple attention heads independently compute attention and their outputs are concatenated.

For each attention head $j \in [h]$, we compute:

$$Q_j = XW_{Q_j}, \quad K_j = XW_{K_j}, \quad V_j = XW_{V_j}$$

where:

- $W_{Q_j}, W_{K_j}, W_{V_j} \in \mathbb{R}^{d \times d}$ are weight matrices for the attention head j.
- X represents the input (either encoder or decoder representation).

Each head computes scaled dot-product attention:

$$A_j = \text{softmax}\left(\frac{Q_j K_j^T}{\sqrt{d}}\right) V_j$$

The outputs of all heads are concatenated:

$$\text{Concat}(A_1, A_2, ..., A_h) \in \mathbb{R}^{n \times (hd)}$$

Finally, we apply a learnable projection matrix W_O to transform back to d:

$$A = MHA(X) = \text{Concat}(A_1, A_2, ..., A_h)W_O$$

where $W_O \in \mathbb{R}^{(hd) \times d}$.

References

[1] Jay Alammar and Maarten Grootendorst. *Hands-on large language models: language understanding and generation.* " O'Reilly Media, Inc.", 2024.

[2] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.