Mathematical Formulation of BERT (Masked Language Modeling)

Qingfeng Liu

August 3, 2025

Notation

- T: Sequence length
- V: Vocabulary size
- d: Dimensionality of hidden states
- L: Number of Transformer encoder layers
- $\mathcal{M} \subset \{1, \ldots, T\}$: Set of masked token positions
- $y_i \in \{0, \ldots, V-1\}$: Ground-truth token index at position $i \in \mathcal{M}$
- $X \in \mathbb{R}^{T \times d}$: Input embeddings with positional encoding
- $H^{(\ell)} \in \mathbb{R}^{T \times d}$: Output of layer ℓ
- $A^{(\ell)} \in \mathbb{R}^{T \times d}$: Attention output of layer ℓ
- $W \in \mathbb{R}^{V \times d}, \quad b \in \mathbb{R}^{V}$: Output projection weights and bias

What is a Token?

- A token is the smallest text unit handled by BERT.
- The vocabulary $\mathcal{V}$ is a fixed-size set (e.g., 30,522 tokens).
- Tokens are based on WordPiece subword units.
- Example: `unbelievable` $\rightarrow$ ["un", "##believ", "##able"]

Token IDs and Embedding Vectors

- Each token is mapped to a vocabulary index: $x_t \in \{0, 1, \ldots, V-1\}$
- A learnable embedding matrix $E \in \mathbb{R}^{V \times d}$ assigns:

$$e_t = E[x_t] \in \mathbb{R}^d$$

- e_t is the vector representation of token x_t

Input Embedding Construction

Token embedding e_t is combined with positional embedding p_t:

$$x_t = e_t + p_t$$

All token embeddings:

$$X = E[x_1, x_2, \ldots, x_T] + P, \quad X \in \mathbb{R}^{T \times d}$$

- P: position embedding matrix (fixed or learnable)

What is the Embedding Space?

- The embedding space is the d-dimensional real vector space $\mathbb{R}^d$
- Each token is represented as a vector $e_t \in \mathbb{R}^d$
- Similar tokens are placed closer together in this space
- This structure is learned during training based on language context

Who Determines the Embedding Space?

- Humans design the dimensions d and vocabulary size V
- The matrix E is randomly initialized (e.g., from $\mathcal{N}(0, \sigma^2 I)$)
- It is optimized via backpropagation using task losses:

$$E_k \leftarrow E_k - \eta \cdot \nabla_{E_k} \mathcal{L}$$

- The space is shaped to reflect semantic similarity through learning

Summary

- The embedding space is not predefined — it is learned through training
- Tokens acquire meaning via their placement in a distributed representation space
- The quality of the embedding space defines BERT's ability to model language

Step 1: Input Embedding

$$X = \mathrm{TokenEmbed}(x_1, \ldots, x_T) + \mathrm{PositionEmbed}(1, \ldots, T)$$

Step 2: Transformer Encoder ($\ell = 1, \ldots, L$)

$$
\begin{aligned}
Q^{(\ell)} &= H^{(\ell-1)} W^{Q(\ell)} \\
K^{(\ell)} &= H^{(\ell-1)} W^{K(\ell)} \\
V^{(\ell)} &= H^{(\ell-1)} W^{V(\ell)} \\
A^{(\ell)} &= \text{softmax}\left(\frac{Q^{(\ell)} K^{(\ell)\top}}{\sqrt{d}} \right) V^{(\ell)} \\
\tilde{H}^{(\ell)} &= \text{LayerNorm}(A^{(\ell)} + H^{(\ell-1)}) \\
H^{(\ell)} &= \text{LayerNorm}(\text{FFN}(\tilde{H}^{(\ell)}) + \tilde{H}^{(\ell)})
\end{aligned}
$$

Feed Forward Network (FFN)

$$\mathrm{FFN}(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

- Applied independently to each token
- Typically: hidden size $d_{\mathrm{ff}} = 4d$

Step 3: Final Output

$$H = H^{(L)} \in \mathbb{R}^{T \times d}$$

- Used for prediction at masked positions

Step 4: Output Layer (Vocabulary Projection)

$$\forall i \in \mathcal{M}, \quad \hat{y}_i = \mathrm{softmax}(WH_i + b) \in \mathbb{R}^V$$

- $\hat{y}_i[k]$: Probability of token k at position i

Step 5: Loss Function (Cross Entropy)

$$\mathcal{L}_{\mathsf{MLM}} = -\sum_{i \in \mathcal{M}} \log \hat{y}_i[y_i]$$

Final Loss Expression

$$\boxed{\mathcal{L}_{\text{BERT}} = -\sum_{i \in \mathcal{M}} \log\left(\text{softmax}(WH_i + b)[y_i]\right)}$$

Terminology Summary

- $H^{(\ell)}$: Hidden state at layer ℓ
- $A^{(\ell)}$: Attention output at layer ℓ
- FFN: Feed Forward Network (2-layer MLP with ReLU)

Basic Formula of Attention

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

- $Q \in \mathbb{R}^{T \times d_k}$: Queries
- $K \in \mathbb{R}^{T \times d_k}$: Keys
- $V \in \mathbb{R}^{T \times d_v}$: Values

Step 1: Inner Product Similarity

$$S = QK^\top \in \mathbb{R}^{T \times T}$$

- Each element $S_{ij} = \langle q_i, k_j \rangle$ is a dot product
- Measures similarity between query q_i and key k_j
- Geometrically: projection of q_i onto key space

Step 2: Softmax Normalization

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$$

- Scaling by $\sqrt{d_k}$ controls sharpness of softmax
- Each row of A becomes a probability distribution
- Represents attention weights assigned to values

Step 3: Weighted Average of Values

$$Z = AV \in \mathbb{R}^{T \times d_v}$$

- Each output vector z_i is a weighted sum of all value vectors
- Weights determined by similarity between q_i and all k_j
- Can be interpreted as a soft "semantic lookup"

Full Mathematical Interpretation

$$\boxed{\text{Attention}(q_i, K, V) = \sum_{j=1}^{T} \mathrm{softmax}_j\left(\frac{q_i^\top k_j}{\sqrt{d_k}}\right) \cdot v_j}$$

- Weighted average of values based on similarity scores
- Nonlinear weighted linear transformation

Geometric Interpretation

- Query q_i is compared with all keys k_j
- Similar keys receive higher attention weights
- Output vector is pulled toward directions of semantically relevant values
- Attention reshapes input space dynamically via similarity structure