

# Week 2

## Comprehensive Guide to BERT Pre-Training

### Contents

<b>1 From Context-Independent to Context-Sensitive</b>	<b>2</b>
1.1 The Limitation of Static Embeddings . . . . .	2
1.2 BERT: The Bidirectional Breakthrough . . . . .	2
<b>2 The BERT Architecture: Theoretical Detail</b>	<b>2</b>
2.1 Input Representation: A Theoretical Expansion . . . . .	2
2.1.1 1. Token Embeddings and Subword Regularization . . . . .	3
2.1.2 2. Positional Embeddings: Learnable vs. Sinusoidal . . . . .	3
2.2 The Transformer Encoder Block . . . . .	3
2.2.1 Self-Attention Mechanism . . . . .	3
<b>3 Pretraining Tasks and Objectives</b>	<b>4</b>
3.1 Task 1: Masked Language Modeling (MLM) . . . . .	4
3.1.1 The Manifold Hypothesis . . . . .	4
3.1.2 The 80-10-10 Rule: A Bias-Variance Tradeoff . . . . .	4
3.2 Task 2: Next Sentence Prediction (NSP) . . . . .	4
<b>4 Dataset Processing</b>	<b>4</b>
4.1 Document-Level Processing . . . . .	5
4.2 Whole Word Masking . . . . .	5
4.3 Dynamic Masking . . . . .	5
<b>5 Implementation Details</b>	<b>5</b>
5.1 The Prediction Heads . . . . .	5
5.1.1 MaskLM Head and Weight Tying . . . . .	5
5.2 GELU: The Stochastic Regularizer . . . . .	6
<b>6 Optimization Landscape</b>	<b>6</b>
6.1 The AdamW Optimizer . . . . .	6
6.2 Warmup and Decay . . . . .	6

# 1 From Context-Independent to Context-Sensitive

The history of word embeddings can be broadly categorized into two eras: the era of static, context-independent embeddings (like Word2Vec and GloVe), and the modern era of dynamic, context-sensitive representations (like BERT and GPT).

## 1.1 The Limitation of Static Embeddings

Models like Word2Vec and GloVe map each word in the vocabulary to a single, static vector  $\mathbf{v}_w \in \mathbb{R}^d$ . This mapping function  $f : w \rightarrow \mathbf{v}_w$  is injective and constant.

This approach suffers fundamentally from the problem of **Polysemy**—the capacity for a word to have multiple related meanings. Consider the word “bank”:

1. “I deposited money at the **bank**.” (Financial Institution)
2. “They sat on the river **bank**.” (Geological Feature)

In a static model, the vector  $\mathbf{v}_{\text{bank}}$  must essentially be a weighted average of these two disparate meanings. This “smearing” of semantic meaning limits the model’s precision.

### Insight: Embeddings as Functions

In static models, an embedding is a **Lookup Table**. In BERT, an embedding is a **Function**.

$$\mathbf{v}_{\text{BERT}}(w_i) = f(w_i | w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_n; \theta)$$

The representation of a token is inextricably linked to the specific instantiation of its neighbors.

## 1.2 BERT: The Bidirectional Breakthrough

BERT (Devlin et al., 2018) combines the best features of ELMo (bidirectionality) and GPT (Transformer architecture).

### Insight: The Bidirectionality Paradox & Denoising

Standard conditional language modeling ( $P(w_t | w_{<t})$ ) makes true bidirectionality impossible because allowing the model to see  $w_{>t}$  would allow it to “cheat” and see the target. BERT circumvents this by reframing the objective. Instead of standard language modeling, BERT performs **Denoising Autoencoding**. It corrupts the input (via masking) and asks the model to reconstruct the original signal. This permits the use of the Transformer **Encoder**, where the self-attention mechanism allows every token to attend to every other token ( $A_{ij} \neq 0, \forall i, j$ ).

# 2 The BERT Architecture: Theoretical Detail

BERT is a multi-layer bidirectional Transformer Encoder.

## 2.1 Input Representation: A Theoretical Expansion

BERT’s input representation is a sum of three embeddings.

$$\mathbf{E}_{\text{input}} = \mathbf{E}_{\text{token}} + \mathbf{E}_{\text{segment}} + \mathbf{E}_{\text{position}} \quad (1)$$

### 2.1.1 1. Token Embeddings and Subword Regularization

BERT uses **WordPiece** tokenization. Unlike BPE (Byte Pair Encoding) which merges the most frequent pairs, WordPiece merges pairs that maximize the **likelihood** of the training data once added to the vocabulary.

#### Mathematical Detail: WordPiece Optimization

Given a corpus  $C$  and a vocabulary  $V$ , WordPiece seeks to select a new symbol  $u, v$  to merge such that the increase in likelihood  $\mathcal{L}$  is maximized:

$$\text{Score}(u, v) = \frac{\text{freq}(uv)}{\text{freq}(u) \times \text{freq}(v)}$$

This metric favors merging pairs where the individual parts are rare but the combination is frequent, effectively capturing meaningful sub-morphemes (roots, suffixes). This solves the **OOV (Out of Vocabulary)** problem by decomposing unknown words into known subwords (e.g., “interstellar” → “inter”, “stellar”).

### 2.1.2 2. Positional Embeddings: Learnable vs. Sinusoidal

The original Transformer used fixed sinusoidal embeddings:

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

BERT, however, uses **Learnable Positional Embeddings**.

- We initialize a matrix  $\mathbf{P} \in \mathbb{R}^{L_{\max} \times H}$ , where  $L_{\max} = 512$ .
- During training, the model learns the optimal vector representation for "Position 1", "Position 2", etc.
- **Implication:** Unlike sinusoidal embeddings, which can theoretically extrapolate to sequence lengths longer than seen during training, learnable embeddings are strictly bound by  $L_{\max}$ . If you feed 513 tokens to BERT, it will fail because it has no vector for position 513.

## 2.2 The Transformer Encoder Block

Each layer  $l$  in BERT applies the following transformations:

### 2.2.1 Self-Attention Mechanism

The core mathematical operation is Scaled Dot-Product Attention.

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V \quad (2)$$

- **Queries ( $Q$ ), Keys ( $K$ ), Values ( $V$ ):** These are projections of the input  $X$  via learnable matrices  $W^Q, W^K, W^V$ .
- **Scaling Factor ( $\sqrt{d_k}$ ):** We divide by the square root of the head dimension ( $d_k = H/A$ ).
- **Why Scale?** As  $d_k$  increases, the dot products  $q \cdot k$  can grow large in magnitude. This pushes the Softmax function into regions where it has extremely small gradients (vanishing gradients), halting learning. Scaling normalizes the variance to 1.

## 3 Pretraining Tasks and Objectives

BERT optimizes a composite loss function:  $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{NSP}}$ .

### 3.1 Task 1: Masked Language Modeling (MLM)

#### 3.1.1 The Manifold Hypothesis

Why does masking work? The **Manifold Hypothesis** states that natural language lies on a low-dimensional manifold within the high-dimensional space of all possible character combinations. By damaging the input (masking) and forcing the model to project it back onto the manifold (predicting the original word), the model must learn the internal structure (grammar, syntax, semantics) that defines the manifold.

#### 3.1.2 The 80-10-10 Rule: A Bias-Variance Tradeoff

The 15% masking strategy includes a subtle refinement:

1. **80% [MASK]:**  $w_i \rightarrow [\text{MASK}]$ . The core learning signal.
2. **10% Random Word:**  $w_i \rightarrow w_{\text{random}}$ .
  - *Theory:* This acts as a regularizer. It introduces noise, forcing the model to rely on *context* rather than blindly trusting the input token. If the model always sees [MASK] at the target position, it might learn to ignore the input embedding at that position entirely. Random injection keeps the model vigilant.
3. **10% Original Word:**  $w_i \rightarrow w_i$ .
  - *Theory:* This aligns the pre-training distribution with the fine-tuning distribution. In fine-tuning, there are no masks. If the model only predicts masked tokens, it may drift away from representing real words correctly. This also nudges the model to bias its prediction towards the observation if the context is ambiguous.

### 3.2 Task 2: Next Sentence Prediction (NSP)

- **Objective:** Predict  $P(\text{IsNext} | [\text{CLS}])$ .
- **Critique:** Subsequent research (RoBERTa, 2019) showed that NSP might be redundant. The NSP task as implemented often boils down to **Topic Prediction** (do these sentences belong to the same article?) rather than true **Coherence Modeling** (does sentence B logically follow A?).
- Despite this, in the original BERT paper, NSP was crucial for improving performance on tasks requiring relationship understanding (like QA and Natural Language Inference).

## 4 Dataset Processing

Efficiently creating the training data is an engineering feat in itself.

## 4.1 Document-Level Processing

BERT inputs are not just sentences; they are "spans" of text packed to length 512.

- **Context Window Packing:** We don't just take two sentences. We take a continuous span of text until we hit the 512 token limit.
- **Sentence A vs Sentence B:** The "sentences" in NSP are actually spans. Span A might contain 3 actual linguistic sentences, and Span B might contain 4.

## 4.2 Whole Word Masking

Standard BERT masks WordPiece tokens independently.

- Example: "unhappiness" → ["un", "happi", "ness"]
- Independent Masking: ["un", "[MASK]", "ness"]

This is easy for the model to solve because "un" and "ness" strongly imply "happi". **Whole Word Masking** (introduced later) mandates that if one subword of a word is masked, *all* subwords of that word must be masked. This forces the model to recover the semantic meaning from the broader context rather than local morphological patterns.

## 4.3 Dynamic Masking

Original BERT performed masking **once** during data preprocessing. The masks were static. If you trained for 10 epochs, the model saw the exact same masks for the exact same sentence 10 times. *RoBERTa* introduced **Dynamic Masking**, where the masking pattern is generated on-the-fly when the data loader yields a batch. This acts as data augmentation, effectively showing the model slightly different puzzles for the same text every epoch.

# 5 Implementation Details

## 5.1 The Prediction Heads

The architecture of the output heads is distinct from the encoder.

### 5.1.1 MaskLM Head and Weight Tying

$$y_{\text{pred}} = \text{Softmax}(\mathbf{W}_{\text{vocab}}(\text{LayerNorm}(\text{GELU}(\mathbf{W}_h \mathbf{h}_{\text{masked}} + \mathbf{b}))))$$

#### Insight: Weight Tying

The matrix  $\mathbf{W}_{\text{vocab}}$  used to project the hidden state back to the vocabulary size (dimension  $H \times V$ ) is often **tied** (shared) with the input embedding matrix.

$$\mathbf{W}_{\text{vocab}} = \mathbf{E}_{\text{token}}^T$$

This significantly reduces the parameter count and forces the input embedding space and output projection space to align.

## 5.2 GELU: The Stochastic Regularizer

BERT uses the **Gaussian Error Linear Unit**. Why not ReLU ( $x > 0?x : 0$ )? ReLU is deterministic based on the sign of  $x$ . GELU can be interpreted as weighting inputs by their magnitude via a stochastic regularizer (dropout). In the limit of discrete noise, it smooths the decision boundary. This curvature helps optimization in deep Transformers where the loss landscape is non-convex.

# 6 Optimization Landscape

## 6.1 The AdamW Optimizer

Standard SGD is rarely used for Transformers. BERT uses AdamW.

- **Adam:** Adaptive Moment Estimation. It keeps running averages of gradients ( $m_t$ ) and squared gradients ( $v_t$ ).
- **The Weight Decay Fix (AdamW):** In standard libraries, L2 regularization is often implemented as adding  $\lambda\theta^2$  to the loss. In adaptive optimizers like Adam, this interacts poorly with the adaptive learning rates. AdamW decouples weight decay, applying it directly to the parameter update step:

$$\theta_t = \theta_{t-1} - \eta(\text{AdamUpdate}) - \eta\lambda\theta_{t-1}$$

This ensures weight decay actually decays the weights, regardless of the gradient magnitude.

## 6.2 Warmup and Decay

The learning rate schedule is critical.

- **Linear Warmup:** For the first 10,000 steps, learning rate increases. This is necessary because, at initialization, the Transformer parameters are random. The gradients are extremely noisy. A high learning rate immediately would destabilize the updates. Warmup allows the optimizer to gather statistics ( $m_t, v_t$ ) and move towards a stable region of the loss landscape before stepping efficiently.

### Things to Ponder

1. **Attention Quadratic Cost:** Self-attention is  $O(N^2)$  in time and memory. For a sequence length  $N = 512$ , this is manageable. For  $N = 4096$ , it becomes prohibitive. How do newer models (Longformer, BigBird) approximate this calculation to handle longer contexts?
2. **Anisotropy in Embeddings:** Research suggests that BERT embeddings suffer from the "representation degeneration problem," where embeddings occupy a narrow cone in the vector space (high anisotropy) rather than being uniform. Why does this happen, and how does LayerNorm contribute to this?
3. **Post-LN vs Pre-LN:** BERT uses Post-LN (LayerNorm *after* the residual connection). Modern LLMs (like Llama) use Pre-LN. Post-LN generally achieves better performance if it converges, but Pre-LN is much more stable during training. Why? (Hint: Consider the gradient flow through the residual path).