Positional Encoding

James Pustejovsky

CS115B - Brandeis University

April 8, 2025

Outline and Goals for Today

1. Motivation and Background
2. The Mathematics of Sinusoidal Encoding
3. Step-by-Step Example: d_model = 4
4. Integration with Transformer Models
5. Technical Details
6. Further Topics

Lecture Objectives

- Understand why self-attention lacks order awareness.
- Learn the details behind sinusoidal positional encoding.
- Derive the mathematical formulas.
- Work through detailed numerical examples.
- Compare fixed (sinusoidal) vs learned approaches.
- Discuss implications for transformer models.

The Problem of Position in Self-Attention

- **Self-attention models** (e.g., Transformers) process all tokens in parallel.
- This design lacks a notion of **sequential order** because it treats inputs as a set.
- **Language is sequential:** The order of words is crucial for syntax and semantics.

Example

Compare the meaning of:

`"The cat sat on the mat"` vs. `"The mat sat on the cat"`

Implicit Positional Information in Other Architectures

- **Recurrent Neural Networks (RNNs):** Process tokens sequentially, inherently preserving order.
- **Convolutional Neural Networks (CNNs):** Use localized filters that capture spatial (or temporal) proximity.
- **Transformers:** Lack these inherent mechanisms, thus requiring explicit injection of positional data.

Introducing Positional Encoding

Definition

Positional encoding adds a vector to each token embedding to convey its position in the sequence.

- It must have the same dimensionality as the token embeddings.
- It encodes *absolute* or *relative* positional information.

Why a Fixed, Sinusoidal Approach?

- **Deterministic:** No additional parameters; the encoding is computed by a fixed function.
- **Generalization:** Can be extrapolated to positions longer than seen during training.
- **Variety:** Different frequencies capture both fine-grained and coarse-grained positions.

Sine and Cosine: Mathematical Properties

- **Periodicity:** $\sin(x)$ and $\cos(x)$ are periodic, providing a repeated, yet unique pattern.
- **Differentiability:** Smooth, continuous functions with well-defined derivatives.
- **Frequency Variation:** Adjusting the input scaling changes the frequency of oscillation.

These properties allow us to generate a unique signature for each position.

The Sinusoidal Positional Encoding Formula

For each position *pos* and model dimension d_{model}, we define:

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

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

where:

- *pos* is the token's position.
- i indexes over the dimensions.

Dissecting the Denominator: $10000^{\frac{2i}{d_{\text{model}}}}$

- This term ensures each dimension oscillates at a different frequency.
- When $i = 0$:
  $$10000^{\frac{0}{d_{\text{model}}}} = 1$$
  so the function uses its natural frequency.
- For higher i, the wavelength increases (frequency decreases).
- This diversity in frequencies allows capturing both short-range and long-range dependencies.

Frequency Perspective of Positional Encoding

- Each dimension $2i$ or $2i + 1$ represents a sine or cosine function with a specific frequency.
- Lower dimensions capture high-frequency variations (fine details).
- Higher dimensions capture low-frequency variations (global structure).

This multiscale representation is key to encoding varied positional information.

Differentiability and Gradient Flow

- Sine and cosine functions are smooth and differentiable, which aids gradient-based learning.
- While the positional encodings themselves are fixed, their smooth variation helps the network learn by providing subtle differences between nearby positions.

Role of Even and Odd Dimensions

- **Even-indexed dimensions** (e.g., $0, 2, 4, \ldots$) use the sine function.
- **Odd-indexed dimensions** (e.g., $1, 3, 5, \ldots$) use the cosine function.

This separation provides two distinct phases of the same underlying wave, enriching the positional signature.

Recap: The Positional Encoding Equations

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

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

- *pos*: token position (0-indexed or 1-indexed, depending on implementation).
- d_{model}: dimensionality of the embeddings.
- i: the index over the half of the dimensions.

Example Setup: $d_{\text{model}} = 4$

- We choose a small embedding size, $d_{\text{model}} = 4$, for clarity.
- We will compute the positional encoding for several positions: $pos = 0$, $pos = 1$, and $pos = 3$.
- Recall: For $i = 0, 1$ (since $2i$ and $2i + 1$ will cover 0 to 3).

Calculating PE at $pos = 0$

- For $i = 0$:

$$\text{PE}(0,0) = \sin\left(\frac{0}{10000^{0/4}}\right) = \sin(0) = 0$$

$$\text{PE}(0,1) = \cos\left(\frac{0}{10000^{0/4}}\right) = \cos(0) = 1$$

- For $i = 1$:

$$\text{PE}(0,2) = \sin\left(\frac{0}{10000^{2/4}}\right) = \sin\left(\frac{0}{100}\right) = 0$$

$$\text{PE}(0,3) = \cos\left(\frac{0}{10000^{2/4}}\right) = \cos(0) = 1$$

Thus, $\text{PE}(0) = [\,0,\ 1,\ 0,\ 1\,]$.

Calculating PE at $pos = 1$ – Even Dimensions

- For $i = 0$:

$$\text{PE}(1,0) = \sin\left(\frac{1}{10000^{0/4}}\right) = \sin(1)$$

  Approximating: $\sin(1) \approx 0.8415$.

- For $i = 1$:

$$\text{PE}(1,2) = \sin\left(\frac{1}{10000^{2/4}}\right) = \sin\left(\frac{1}{100}\right)$$

  For small angle, $\sin(0.01) \approx 0.01$.

Calculating PE at $pos = 1$ – Odd Dimensions

- For $i = 0$:

$$\text{PE}(1,1) = \cos(1)$$

Approximating: $\cos(1) \approx 0.5403$.

- For $i = 1$:

$$\text{PE}(1,3) = \cos\left(\frac{1}{100}\right) = \cos(0.01)$$

For small angles, $\cos(0.01) \approx 0.99995$.

Thus, $\text{PE}(1) \approx [0.8415,\ 0.5403,\ 0.01,\ 0.99995]$.

Calculating PE at $pos = 3$: Even Dimensions

- For $i = 0$:
$$\text{PE}(3, 0) = \sin(3) \quad (\text{since } 10000^{0/4} = 1)$$
Approximate: $\sin(3) \approx 0.1411$.
- For $i = 1$:
$$\text{PE}(3, 2) = \sin\left(\frac{3}{100}\right) = \sin(0.03)$$
Approximate: $\sin(0.03) \approx 0.03$.

Calculating PE at $pos = 3$: Odd Dimensions

- For $i = 0$:

$$\text{PE}(3, 1) = \cos(3)$$

Approximate: $\cos(3) \approx -0.9899$.

- For $i = 1$:

$$\text{PE}(3, 3) = \cos\left(\frac{3}{100}\right) = \cos(0.03)$$

Approximate: $\cos(0.03) \approx 0.99955$.

So, $\text{PE}(3) \approx [0.1411,\ -0.9899,\ 0.03,\ 0.99955]$.

Step-by-Step: Recap for $pos = 3$

For $d_{\textbf{model}} = 4$:

1. **Dimension 0** :

$$PE(3,0) = \sin\left(\frac{3}{1}\right) = \sin(3) \approx 0.1411.$$

2. **Dimension 1** :

$$PE(3,1) = \cos(3) \approx -0.9899.$$

3. **Dimension 2** :

$$PE(3,2) = \sin\left(\frac{3}{100}\right) = \sin(0.03) \approx 0.03.$$

4. **Dimension 3** :

$$PE(3,3) = \cos(0.03) \approx 0.99955.$$

Graphical Illustration of Sinusoidal Curves

- Plotting $\sin(x)$ and $\cos(x)$ shows smooth, periodic oscillations.
- Different scaling factors (e.g., x vs. $x/100$) yield curves with different wavelengths.

[Insert Plot: Multiple sine/cosine curves demonstrating frequency changes]

Frequency and Distinct Positional Signatures

- Each dimension's scaling factor determines its frequency.
- High-frequency components (lower i) capture rapid positional changes.
- Low-frequency components (higher i) capture broad, global position information.

Implication: The combination across dimensions results in a unique, multi-scale encoding.

Advantages of Sinusoidal Positional Encoding

- **Parameter-free:** No extra learnable parameters are introduced.
- **Extrapolation:** Functions generalize to positions beyond training.
- **Smooth Variation:** Nearby positions yield similar encodings—useful for learning relative distances.
- **Dual Functions:** Use of sine and cosine provides complementary phase information.

Integrating Positional Encoding with Token Embeddings

- Token embeddings: Represent the meaning of the token.
- Positional encodings: Provide information about the token's position.
- **Combined Representation:**

$$\text{Input Representation} = \text{Token Embedding} + \text{Positional Encoding}$$

This addition preserves the embedding dimension and introduces order-sensitive information.

Mathematics of Combined Embeddings

Let:

$$\mathbf{E}_t \in \mathbb{R}^{d_{\text{model}}} \quad \text{be the embedding of token at position } t,$$

and

$$\mathbf{P}_t \in \mathbb{R}^{d_{\text{model}}} \quad \text{be the positional encoding for position } t.$$

Then the input to the Transformer is:

$$\mathbf{X}_t = \mathbf{E}_t + \mathbf{P}_t.$$

This summation ensures both semantic and positional information are available to the self-attention mechanism.

Influence on Self-Attention Computations

- Self-attention computes dot products between queries and keys.
- With positional encoding, these dot products include contributions from both content and position.
- This helps the model differentiate otherwise similar tokens that occur at different positions.

Backpropagation Through Positional Encodings

- As the sinusoidal encoding is fixed (not learned), no gradients are computed for these encodings.
- The gradients flow only through the token embeddings and subsequent layers.
- This simplicity avoids potential issues with overfitting on position.

Fixed vs. Learned Positional Embeddings

- **Fixed (Sinusoidal):**
  - No additional parameters.
  - Extrapolates naturally to longer sequences.
- **Learned:**
  - Parameters are updated during training.
  - May not extrapolate well beyond training positions.

Trade-Off: Fixed encodings are simple and robust, while learned encodings offer flexibility at the cost of increased parameterization.

Why the $\frac{pos}{10000^{\frac{2i}{d_{\text{model}}}}}$ Term?

- The power term $\frac{2i}{d_{\text{model}}}$ ensures the wavelengths form a geometric progression.
- This progression guarantees that each dimension represents a different frequency scale.
- Using 10000 as the base is an empirical choice, large enough to cover a wide range of positions.

Differentiation of the Sine Component

Consider the derivative with respect to *pos* for the even-dimension:

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

- This shows how a change in position affects the encoding.
- Similar derivation holds for the cosine components.

Understanding Wavelength and Phase

- **Wavelength:** Determined by $10000^{\frac{2i}{d_{\text{model}}}}$; larger for higher dimensions.
- **Phase:** Sine and cosine functions differ by a phase shift ($\pi/2$), providing complementary information.
- Together, they allow the model to discern fine-grained and coarse positional differences.

Extended Example: $d_{\text{model}} = 8$ for $pos = 3$

- Now consider a model with $d_{\text{model}} = 8$.
- For each $i = 0, 1, 2, 3$, compute:

$$\text{PE}(3, 2i) = \sin\left(\frac{3}{10000^{\frac{2i}{8}}}\right)$$

$$\text{PE}(3, 2i+1) = \cos\left(\frac{3}{10000^{\frac{2i}{8}}}\right)$$

- The first pair ($i = 0$) is identical to our previous computations; higher i yield different scaling.

Computational Efficiency Considerations

- The fixed nature of sinusoidal encodings requires minimal additional computation.
- Memory overhead is small since the encoding can be computed on the fly.
- Efficient for both training and inference, especially when extrapolating to longer sequences.

Influence on Attention Weights

- The attention mechanism computes scores as:

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

- With combined embeddings, the dot products incorporate position.
- This can modulate the attention scores based on the relative positions of tokens.

Empirical Benefits in Transformer Models

- Studies show that positional encoding improves performance on tasks like translation, summarization, and language modeling.
- The sinusoidal method is especially beneficial in scenarios where sequence lengths vary widely.
- Research continues to explore alternatives (e.g., relative positional encoding) to further enhance performance.

Relative vs Absolute Positional Encodings

- **Absolute Encodings:** Fixed positions provided by sinusoidal or learned vectors.
- **Relative Encodings:** Focus on the distance between tokens.
- Relative encodings can offer better performance for some tasks by directly modeling inter-token distance.

Future Directions and Open Questions

- How do modifications in the base (10000) or the exponent affect learning?
- Can hybrid approaches combining fixed and learned components yield improvements?
- What tasks benefit the most from relative positional encodings versus absolute ones?

Summary and Recap

- **Problem:** Self-attention lacks inherent positional information.
- **Solution:** Add sinusoidal positional encodings using:

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right),$$
$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right).$$

- **Example:** Detailed computations for $d_{\mathrm{model}} = 4$ and $d_{\mathrm{model}} = 8$.
- **Integration:** Combined with token embeddings to inform the attention mechanism.