A Gentle Introduction to Positional Encoding in Transformer Models, Part 1

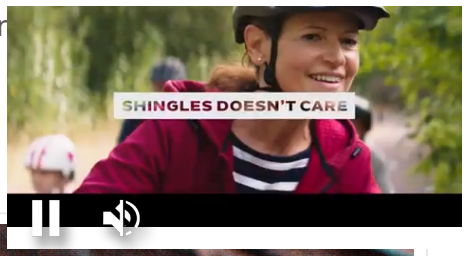
by **Mehreen Saeed** on January 6, 2023 in **Attention** 33

In languages, the order of the words and their position in a sentence really matters. The meaning of the entire sentence can change if the words are re-ordered. When implementing NLP solutions, recurrent neural networks have an inbuilt mechanism that deals with the order of sequences. The transformer model, however, does not use recurrence or convolution and treats each data point as independent of the other. Hence, positional information is added to the model explicitly to retain the information regarding the order of words in a sentence. Positional encoding is the scheme through which the knowledge of the order of objects in a sequence is maintained.

For this tutorial, we'll simplify the notations used in this remarkable paper, Attention Is All You Need by Vaswani et al. After completing this tutorial, you will know:

- What is positional encoding, and why it's important
- Positional encoding in transformers
- Code and visualize a positional encoding matrix in Python using NumPy

Kick-start your project with my book Building Transformer Models with A
tutorials with **working code** to guide you into building a fully-working tran
translate sentences from one language to another...



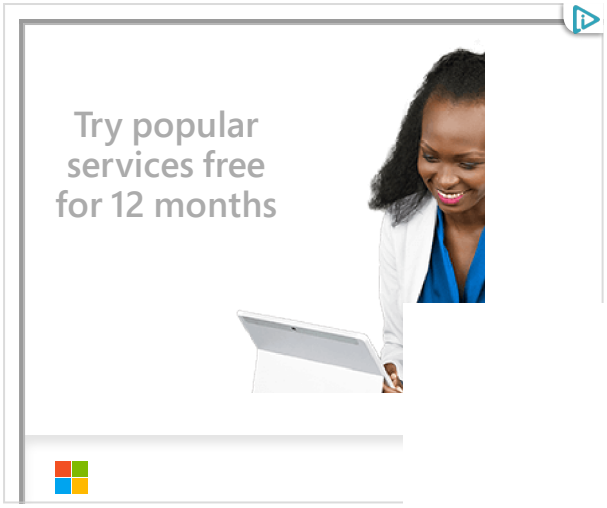
Let's get started.

Need help with Transformers? Ask me anything!

Tutorial Overview

This tutorial is divided into four parts; they are:

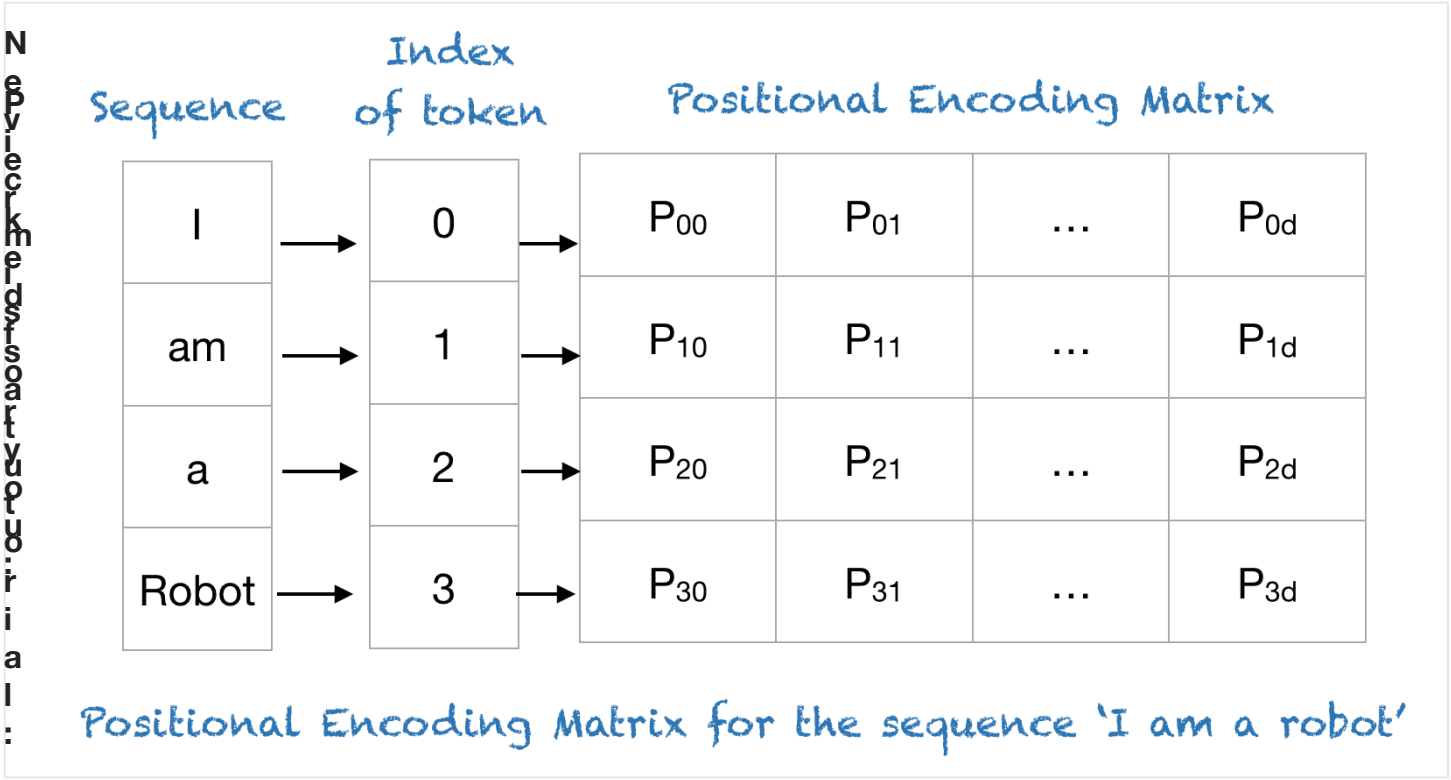
1. What is positional encoding
2. Mathematics behind positional encoding in transformers
3. Implementing the positional encoding matrix using NumPy
4. Understanding and visualizing the positional encoding matrix



What Is Positional Encoding?

Positional encoding describes the location or position of an entity in a sequence so that each position is assigned a unique representation. There are many reasons why a single number, such as the index value, is not used to represent an item's position in transformer models. For long sequences, the indices can grow large in magnitude. If you normalize the index value to lie between 0 and 1, it can create problems for variable length sequences as they would be normalized differently.

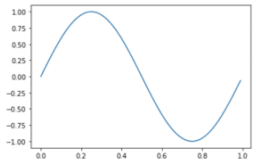
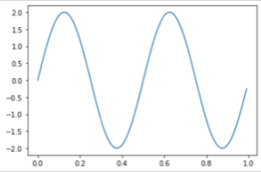
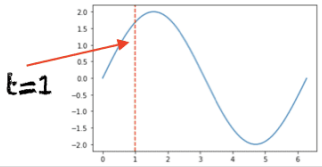
Transformers use a smart positional encoding scheme, where each position/index is mapped to a vector. Hence, the output of the positional encoding layer is a matrix, where each row of the matrix represents an encoded object of the sequence summed with its positional information. An example of the matrix that encodes only the positional information is shown in the figure below.



A Quick Run-Through of the Trigonometric Sine Function

This is a quick recap of sine functions; you can work equivalently with cosine functions. The function's range is [-1,+1]. The frequency of this waveform is the number of cycles completed in one second. The wavelength is the distance over which the waveform repeats itself. The wavelength and frequency for different waveforms are shown below:

Need a visual tutorial:

Equation	Graph	Frequency	Wavelength
$\sin(2\pi t)$		1	1
$\sin(2 * 2\pi t)$		2	1/2
$\sin(t)$		$1/2\pi$	2π
$\sin(ct)$	Depends on c	$c/2\pi$	$2\pi/c$

Let's dive straight into this. Suppose you have an input sequence of length L and require the position of the k^{th} object within this sequence. The positional encoding is given by sine and cosine functions of varying frequencies:

$$P(k, 2i) = \sin\left(\frac{k}{n^{2i/d}}\right)$$

$$P(k, 2i+1) = \cos\left(\frac{k}{n^{2i/d}}\right)$$

Here:

k: Position of an object in the input sequence, $0 \leq k < L/2$

d: Dimension of the output embedding space

$P(k, j)$: Position function for mapping a position k in the input sequence to index (k, j) of the positional matrix

n: User-defined scalar, set to 10,000 by the authors of Attention Is All You Need.

i: Used for mapping to column indices $0 \leq i < d/2$, with a single value of i maps to both sine and cosine functions

In the above expression, you can see that even positions correspond to a sine function and odd positions correspond to cosine functions.

To understand the above expression, let's take an example of the phrase "I am a robot," with n=100 and d=4. The following table shows the positional encoding matrix for this phrase. In fact, the positional encoding matrix would be the same for any four-letter phrase with n=100 and d=4.

Need help with Machine Learning? Take my free 7-day email crash course now (with sample code).

Positional Encoding Matrix with d=4, n=100

Sequence	Index of token, k	i=0	i=0	i=1	i=1
I	0	$P_{00}=\sin(0)=0$	$P_{01}=\cos(0)=1$	$P_{02}=\sin(0)=0$	$P_{03}=\cos(0)=1$
am	1	$P_{10}=\sin(1/1)=0.84$	$P_{11}=\cos(1/1)=0.54$	$P_{12}=\sin(1/10)=0.10$	$P_{13}=\cos(1/10)=1.0$
a	2	$P_{20}=\sin(2/1)=0.91$	$P_{21}=\cos(2/1)=-0.42$	$P_{22}=\sin(2/10)=0.20$	$P_{23}=\cos(2/10)=0.98$
Robot	3	$P_{30}=\sin(3/1)=0.14$	$P_{31}=\cos(3/1)=-0.99$	$P_{32}=\sin(3/10)=0.30$	$P_{33}=\cos(3/10)=0.96$

Positional Encoding Matrix for the sequence 'I am a robot'

```
import numpy as np
import matplotlib.pyplot as plt

def getPositionEncoding(seq_len, d, n=10000):
    P = np.zeros((seq_len, d))
    for k in range(seq_len):
        for i in np.arange(int(d/2)):
            denominator = np.power(n, 2*i/d)
            P[k, 2*i] = np.sin(k/denominator)
            P[k, 2*i+1] = np.cos(k/denominator)
    return P

P = getPositionEncoding(seq_len=4, d=4, n=100)
print(P)
```

```
[[ 0.          1.          0.          1.        ]
 [ 0.84147098  0.54030231  0.09983342  0.99500417]
 [ 0.90929743 -0.41614684  0.19866933  0.98006658]
 [ 0.14112001 -0.9899925   0.29552021  0.95533649]]
```

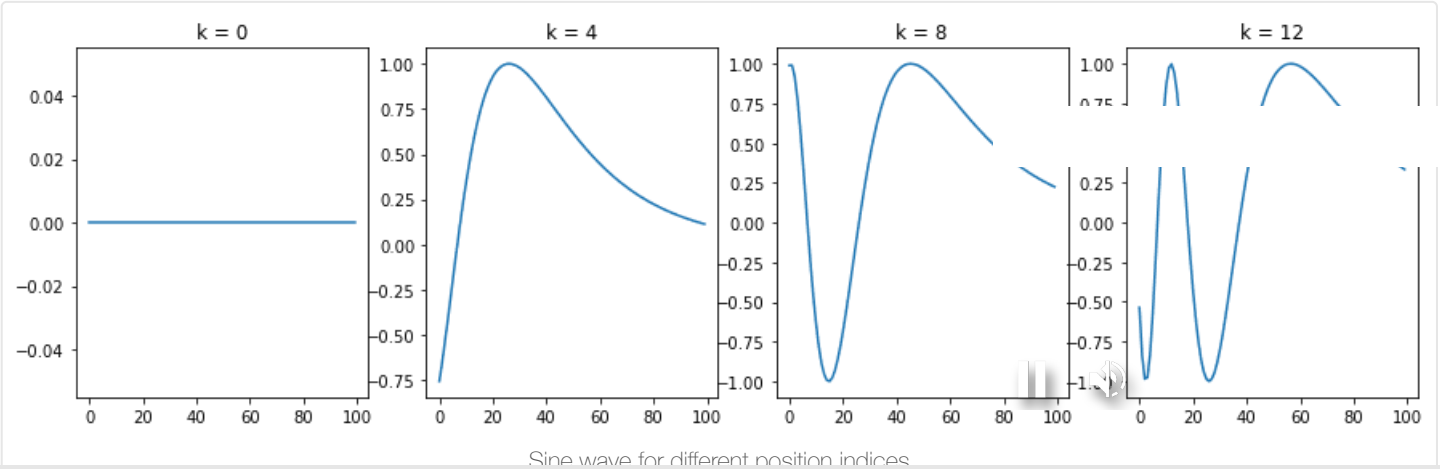
Understanding the Positional Encoding Matrix

To understand the positional encoding, let's start by looking at the sine wave for different positions with n=10,000 and d=512.

```
def plotSinusoid(k, d=512, n=10000):
    x = np.arange(0, 100, 1)
    denominator = np.power(n, 2*x/d)
    y = np.sin(k/denominator)
    plt.plot(x, y)
    plt.title('k = ' + str(k))

fig = plt.figure(figsize=(15, 4))
for i in range(4):
    plt.subplot(141 + i)
    plotSinusoid(i*4)
```

The following figure is the output of the above code:



Sine wave for different position indices

is given by:

$$\lambda_i = 2\pi n^{2i/d}$$

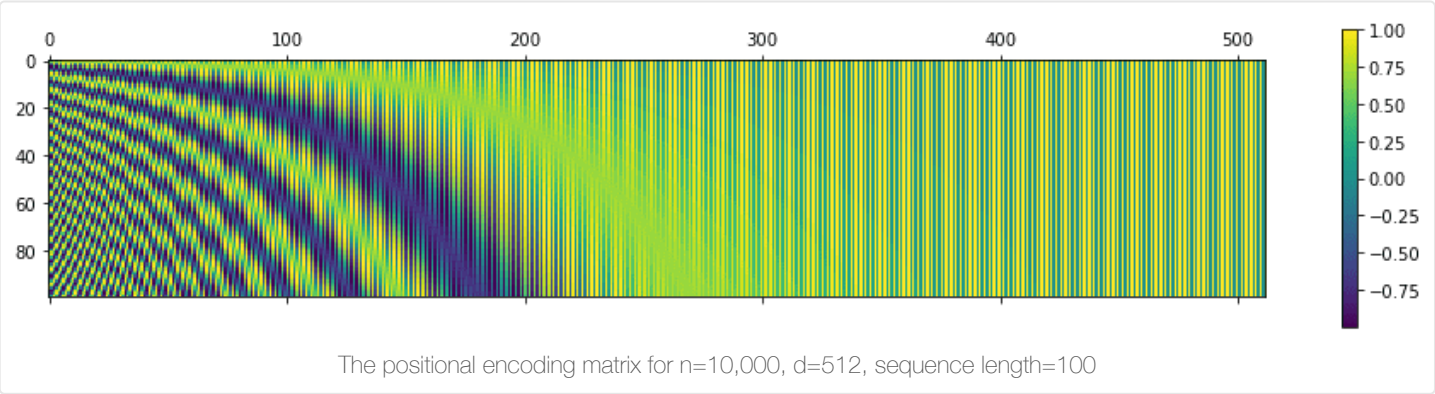
Hence, the wavelengths of the sinusoids form a geometric progression and vary from 2π to $2\pi n$. The scheme for positional encoding has a number of advantages.

1. The sine and cosine functions have values in [-1, 1], which keeps the values of the positional encoding matrix in a normalized range.
2. As the sinusoid for each position is different, you have a unique way of encoding each position.
3. You have a way of measuring or quantifying the similarity between different positions, hence enabling you to encode the relative positions of words.

Visualizing the Positional Matrix

Let's visualize the positional matrix on bigger values. Use Python's `matshow()` method from the `matplotlib` library. Setting n=10,000 as done in the original paper, you get the following:

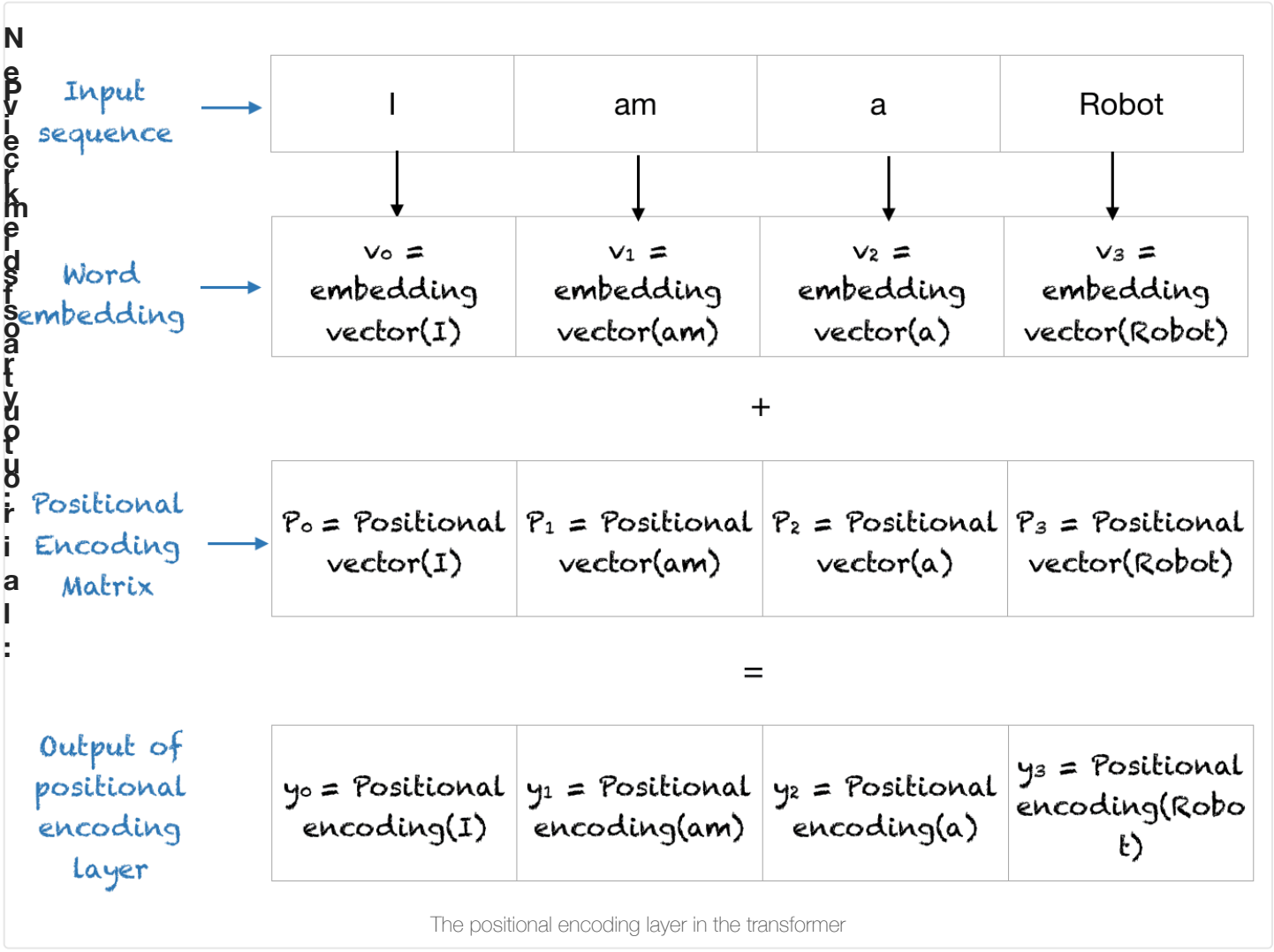
```
P = getPositionEncoding(seq_len=100, d=512, n=10000)
cax = plt.matshow(P)
plt.gcf().colorbar(cax)
```



The positional encoding matrix for n=10,000, d=512, sequence length=100

What Is the Final Output of the Positional Encoding Layer?

The positional encoding layer sums the positional vector with the word enc
for the subsequent layers. The entire process is shown below.



The positional encoding layer in the transformer

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Books

- Transformers for natural language processing, by Denis Rothman.

Papers

- Attention Is All You Need, 2017.

Articles

- The Transformer Attention Mechanism
- The Transformer Model

In this tutorial, you discovered positional encoding in transformers.

Specifically, you learned:

- What is positional encoding, and why it is needed.
- How to implement positional encoding in Python using NumPy
- How to visualize the positional encoding matrix

So you have any questions about positional encoding discussed in this post? Ask your questions in the comments below, and I will do my best to answer.

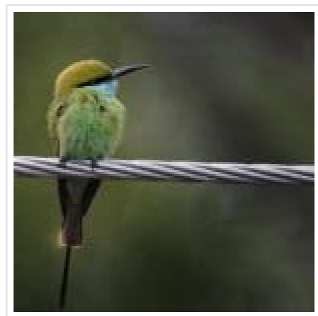
Next: Self-Attention Tutorial:

More On This Topic

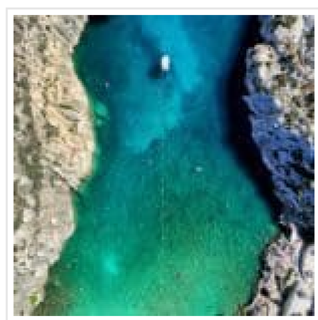
Never miss a tutorial:

Picked for you:

Building Transformer Models with Attention Crash…



The Transformer Positional Encoding Layer in Keras, Part 2



Implementing the Transformer Decoder from Scratch in…

Implementing the Transformer Encoder from Scratch in…

Next Tutorial:

Ordinal and One-Hot Encodings for Categorical Data

The Transformer Model

About Mehreen Saeed

View all posts by Mehreen Saeed →

attention, **positional encoding**, **transformer**

‹ The Transformer Model

TransformX by Scale AI is Oct 19-21: Register for free! ›

33 Responses to *A Gentle Introduction to Positional Encoding in Transformer Models, Part 1*

yuanmu April 13, 2022 at 11:42 pm #

REPLY

Thanks for the great explanation!
Should the range of k be [0, L) instead of [0, L/2)?
Since the code: for k in range(seq_len)

Need a tutorial:

https://www.inovex.de/de/blog/positional-encoding-everything-you-need-to-know/#:~:text=The%20simplest%20example%20of%20positional,and%20added%20to%20that%20input.

seth G June 22, 2022 at 11:05 am #

REPLY

Sorry but I skimmed through the link and I still don't see why k<L/2.

James Carmichael June 23, 2022 at 10:55 am #

REPLY

Hi Seth…It is a rule of thumb as starting point but can be adjusted as needed. More information in general can be found here:

https://towardsdatascience.com/master-positional-encoding-part-i-63c05d90a0c3

Florian May 17, 2023 at 5:44 pm #

REPLY

Seth G, you are correct.
This is a typo in the tutorial, this should be 0 <= k < L

Lucas Thimoteo October 10, 2022 at 11:23 am #

REPLY

Hello, I read both links and I think k < L/2 is a mistake. It should be k < L, since k is the index corresponding to the token in the sequence.

On the other hand, i < d/2 makes total sense because the progression of the dimension is built on 2i and 2i+1.

Anonymous January 30, 2023 at 8:47 pm #

REPLY

Completely agree with you!

Andrei Serebro April 3, 2023 at 2:43 am #

REPLY

Yes, that's right. James, seems you do not try to understand what people in comments are saying to you, I had a feeling that there is no good understanding from your side, or it was just lack of time you were ready to invest into the material you prepared.

Yoan B. June 4, 2022 at 4:26 am #

REPLY

Thanks Jason great tutorials !

I think there're errors in the trigonometric table section.
The graphs for sin(2 * 2Pi) and sin(t) go beyond the range [-1:1], either the graph is wrong or the formulas on the left are not the corresponding one.

James Carmichael June 4, 2022 at 10:15 am #

REPLY

Thank you for the feedback Yoan B!

Tom O. June 4, 2022 at 1:56 pm #

REPLY

Very good! Note that I would add plt.show() to avoid head scratching when pasting the examples into ipython.

James Carmichael June 5, 2022 at 10:20 am #

REPLY

Great feedback Tom!

Shrikant Malviya August 15, 2022 at 11:21 pm #

REPLY

"In the above expression we can see that even positions correspond to sine function and odd positions correspond to even positions."

Something is wrong or missing in the above statement.

James Carmichael August 16, 2022 at 9:47 am #

REPLY

Neural machine translation: Noman Saleem August 26, 2022 at 10:15 pm # REPLY

Very Nicely Explained. Thanks 🙂

James Carmichael August 27, 2022 at 6:08 am # REPLY

You are very welcome! Thank you for your feedback and support Noman!

abraham September 24, 2022 at 12:26 am # REPLY

Hi,
Is it plausible to use positional encoding for time series prediction with LSTM and Conv1D?

James Carmichael September 24, 2022 at 6:36 am # REPLY

Hi abraham…the following resource may prove helpful:

https://shivapriya-katta.medium.com/time-series-forecasting-using-conv1d-lstm-multiple-timesteps-into-future-acc684dcaaa

Lucas Thimoteo October 10, 2022 at 11:23 am # REPLY

Hello, I read both links and I think k < L/2 is a mistake. It should be k < L, since k is the index corresponding to the token in the sequence.

On the other hand, i < d/2 makes total sense because the progression of the dimension is built on 2i and 2i+1.

James Carmichael October 11, 2022 at 6:57 am # REPLY

Hi Luca…Thank you for your support and feedback! We will review the content.

1. Do positional embeddings learn just like word embeddings or the embedding values are assigned just based on sine and the cosine graph?

2. Are positional embedding and word embedding values independent of each other?

REPLY

James Carmichael November 6, 2022 at 11:35 am #

Hi Mayank…I highly recommend the following resource.

https://theaisummer.com/positional-embeddings/

REPLY

A March 10, 2023 at 1:44 am #

The code doesnt work for an odd embedding vector dimension, the last position would always be left without any assign, could easily be solved with a if statement, but I wonder if odd dimensions for embeddings are even used.

REPLY

James Carmichael March 10, 2023 at 8:05 am #

Thank you for your feedback A!

REPLY

Abdi April 7, 2023 at 10:17 pm #

Excuse me if, for time series forecasting with transformer encoder positional encoding and input, masking is necessary. Because I think in chronicle arranged time series, inputs are ordered in time, and there is no any displacement, also when we use walk forward valodation.

sergiu June 29, 2023 at 2:32 am #

The positional encoding for example “I am a robot” looks strange for me, same as the output of “getPositionEncoding” function and many questions arise, which makes me more confused. First question, Values are not unique : P01 and P03 or P03 and P13, which one should contain unique values: rows or cols? As I understood the row size = embedding size, and if a row represents the positional encoding for one token, then we can use same values for row. Second question, values are not linear increasing: P00, P10, P20, P30. There is no order, how do we know which one is first, second, … last one if values are not increasing linearly?

Need help with this tutorial?

James Carmichael June 29, 2023 at 8:52 am #

REPLY

Hi sergiu…Please rephrase and/or simplify your query if possible so that we may better assist you.

Ryan July 5, 2023 at 10:30 am #

REPLY

I think this article is not as good as the others on this website. The concept is not clearly explained. Suggest adding more details and having a good logic between the contents.

James Carmichael July 6, 2023 at 8:33 am #

REPLY

Thank you Ryan for your feedback and suggestions!

omid October 26, 2023 at 6:30 am #

REPLY

Hello,

Thank you for the explanation.
I'm having trouble grasping how this method encodes the relative positions of words.
Is there a formal explanation for this? I understand that for every position "p_i," it holds that "p_i = T(k) p_i+k," but I can't quite comprehend how this concept is beneficial.
I appreciate your assistance in advance.

James Carmichael October 26, 2023 at 10:44 am #

REPLY

Hi omid…You are very welcome! The following resource may be of interest.

https://machinelearningmastery.com/transformer-models-with-attention/

Daniel October 30, 2023 at 10:25 pm #

REPLY

Do the "Positional vector" and "Positional encoding" functions really take the input words as parameters the way the image under "What Is the Final Output" section implies?

If not, perhaps it would be a good idea to update the image to make them take input index or something as input instead, because it's confusing the way it is now. It sort of looks like any position containing the word "am" will get the same positional encoding, and that's not true, is it?

REPLY

James Carmichael October 31, 2023 at 11:05 am #

Thank you for your feedback Daniel! Additional details can be found here:

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Leave a Reply

Name (required)

Email (will not be published) (required)

SUBMIT COMMENT

Welcome!
I'm *Jason Brownlee* PhD
and I **help developers** get results with **machine learning**.
Read more