Positional Embeddings

aka positional encodings

Table of Contents

01 Positional Embeddings

02 Transformers and need of positional embeddings

03 Requirements of a good Positional Embedding

04 Absolute Positional Embeddings

05 Relative Positional Embeddings

06 Rotary Positional Embedding

07 Learned Positional Embedding

08 References

Introduction

Positional embeddings play a crucial role in the world of Natural Language Processing (NLP), particularly in the context of transformer-based models. These models have revolutionized NLP tasks but lack inherent knowledge of word order, necessitating the incorporation of positional information. In this comprehensive set of notes, we will delve into the fundamental importance of positional embeddings in transformers. We will explore the requirements for effective positional embeddings, including the widely used sine and cosine-based absolute positional encodings, and delve into relative positional encodings, which are vital for understanding the contextual relationships between tokens. Additionally, we will touch upon the innovative concept of rotary positional embeddings that have emerged to further enhance the capabilities of these models.

01 Positional Embedding

Definitions and intuition

Positional Embedding

- **Positional embedding** is used to **provide the positional information of the input** to the non-recurrent architecture of multi-head attention.

- **In NLP** - **represent relative or absolute positions of words in a sentence**.
- also used in Time series data, DNA sequences, Graphs etc.

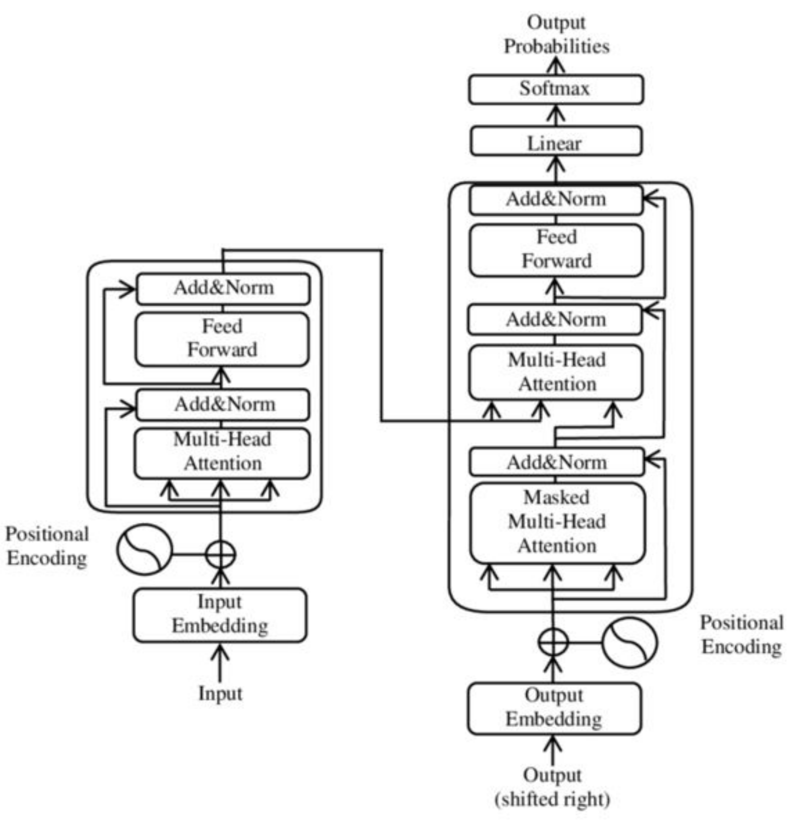
- **Integrating** - **add the tensor (of same shape as input)** that contains the relevant information to the input sequence. (add vs concatanate discussed later)

02 | Transformers and need of Positional Embeddings

Positional embedding in Transformers

- In Recurrent neural networks (**RNNs**) , **input sequential order implicitly defined**.

- However, In **Transformer's** Multi-Head Attention layer **processes the entire sequence simultaneously**, which means it lacks inherent knowledge of the order. It treats each element in the sequence independently, which can result in the **loss of context related to their order**.
- This issue also applies to **convolutional layers,** which have a **limited local context** for sequential ordering determined by their kernel size, making them less suitable for tasks that require capturing long-range dependencies in the data.



03 | Requirements of a good Positional Embedding

Challenges and requirements of Positional Embeddings

- Varying sequence length is challenge encountered here.
  Trials: Absolute index (size could go very high) , normalized index (meaning changes with varying length) etc.
  Need:
  - **Input length independent**
  - **Input content independent**
  - **Unique embedding**

- The positional informations should not be prioritized over the semantics, that could defeat the purpose.
  Need:
  - **small values** (relative to input)

04 Absolute Positional Embeddings

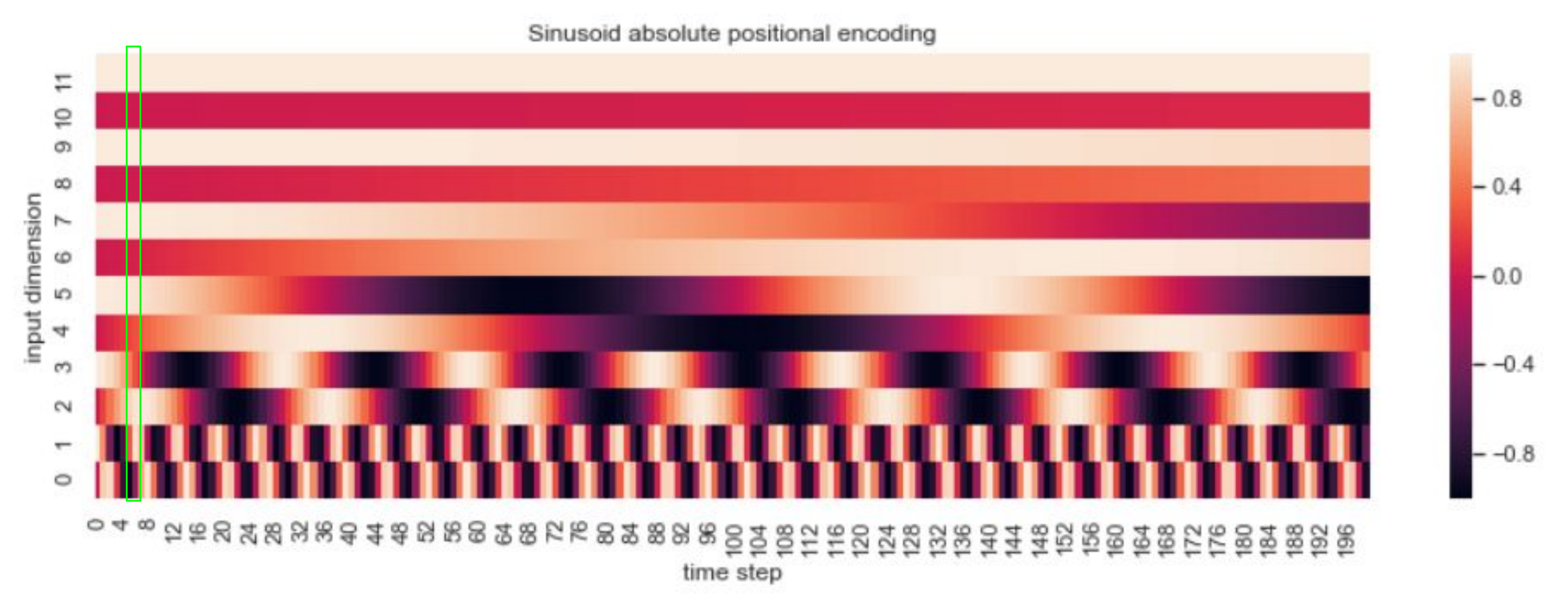
Absolute positional embedding

- **Absolute positional encoding** assigns a unique label to each position in a sequence **based on its distance from the beginning** of that sequence, creating a global coordinate system relative to the sequence's start.
- Trials:
  - **Absolute index** (large values for longer sequences)
  - **normalized index** (length dependent embedding)
  -
- Authors of [Vaswani et al., 2017] however proposed a different absolute **positional encoding based on the sine and cosine functions**:

$$PE_{(\mathrm{pos},2i)} = \sin(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}})$$
$$PE_{(\mathrm{pos},2i+1)} = \cos(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}})$$

pos = position in time/sequence
d_{model} = dimension index

Sinusoid absolute positional embedding



Credits:
https://www.inovex.de/de/blog/positional-encoding-everything-you-need-to-know/

Challenge

- learning difference for different positions:
  - Even if the dataset had an equal distribution of sequence lengths, the model would still **favor the initial positions**, because shorter sequences appear more frequently during training.
  - Potentially leads to **suboptimal performance for longer sequences** during testing or real-world use.

- Add vs concatenate embedding:
  - **Adding** : **need balance** between input info and positional info)
  - **Concatenating** (**need more computational expense** -extra dimensions)

Following the initial research, a subsequent study by [Shaw et al. , 2018] addressed the challenge of adapting to various input sequence lengths. In this study, they introduced **relative positional encoding as a solution** to this issue.

05 Relative Positional Embeddings

Relative Positional Embeddings

- **Relative Positional Embeddings** are based on the **relative distance between the elements in sequence** (or graph etc), rather than their absolute order.
- Rather than a single positional embedding for an element in sequence, here they can have as **many embeddings** as the total number of elements in the sequence, **each characterizing the relative positional relationship**. (including itself)
- Here, (as in [Shaw et al. , 2018]), the relation to itself is taken as w_0 and w_{+1}, w_{+2} to the right and w_{-1}, w_{-2} to the left.
- These embeddings, **(……w_{-1}, w_{-2}, w_0, w_{-1}, w_{-2} ,……)** doesnt change its value regardless of the element it is representing. This takes care of the sequence length as well as the requirement for the good positional embedding.

- Additionally, they have also done **clipping**, in which after certain distance, the positional embedding remain the same, reducing the total number of embeddings required.
- For Integrating this, we **modify the self attention**: [$a_{(i,j)} = a_{(pos,\ related\ pos)}$]

Self-Attention

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ij}}$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K)^T}{\sqrt{d_z}}$$

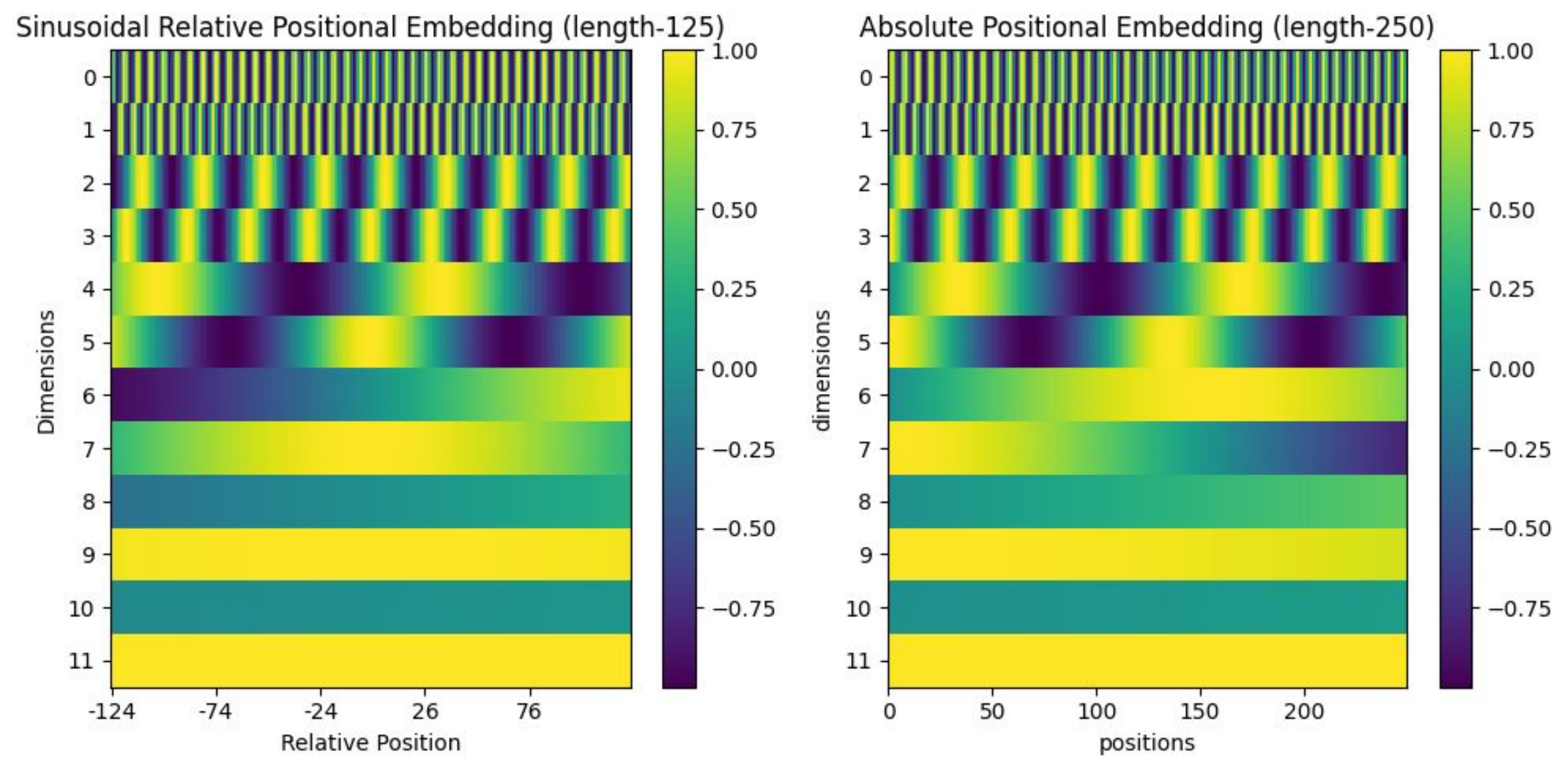
Relation-aware Self-Attention

$$z_i = \sum_{j=1}^{n} \alpha_{ij}(x_j W^V + a_{ij}^V)$$

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ij}}$$

$$e_{ij} = \frac{x_i W^Q (x_j W^K + a_{ij}^K)^T}{\sqrt{d_z}}$$

Comparison: Absolute vs relative (both sinusoidal)



Features of the sinusoidal relative embedding

- **Uniqueness**:
    - Since its not symmetric through origin, the relative positions to left and right can be differentiated.

- **Input length independence**:
    - SInce the relative positions are always from the origin, the embeddings arent dependent on input length.

05 | Rotary Positional Embeddings

Rotary Positional Embeddings (RoPE)

- Relative positional embeddings are slower due to the additional step in the self attention layer. **RoPE** is a combination of relative and absolute positional embedding ideas. [Su, Jianlin et al. 2021]. The **embeddings are applied with rotation** depending upon their distance from the beginning.
- For representation, consider an element in sequence represent in 2 dimension, the rotary embedding rotates the element embedding by ***(m*theta)***

  m = position

  Theta = fixed small angle
- The **relative position is conserved** here, as the relative rotations angles corresponds to the relative distances of the words. Adding words before or after doesn't affect it.

  E.g. **He** is **Adhil**. (relative angle: **2*theta**)

  They know that **he** is **adhil.** (relative angle: **2*theta**)

- The **example** case equation.

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix} \begin{pmatrix} W^{(11)}_{\{q,k\}} & W^{(12)}_{\{q,k\}} \\ W^{(21)}_{\{q,k\}} & W^{(22)}_{\{q,k\}} \end{pmatrix} \begin{pmatrix} x^{(1)}_m \\ x^{(2)}_m \end{pmatrix}$$

- In **general dimensions**:

$$f_{\{q,k\}}(\boldsymbol{x}_m, m) = \boldsymbol{R}^d_{\Theta,m} \boldsymbol{W}_{\{q,k\}} \boldsymbol{x}_m$$

where

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$

$$\Theta = \{\theta_i = 10000^{-2(i-1)/d}, i \in [1, 2, ..., d/2]\}.$$

- **Computer efficient** general form of Rotational matrix:

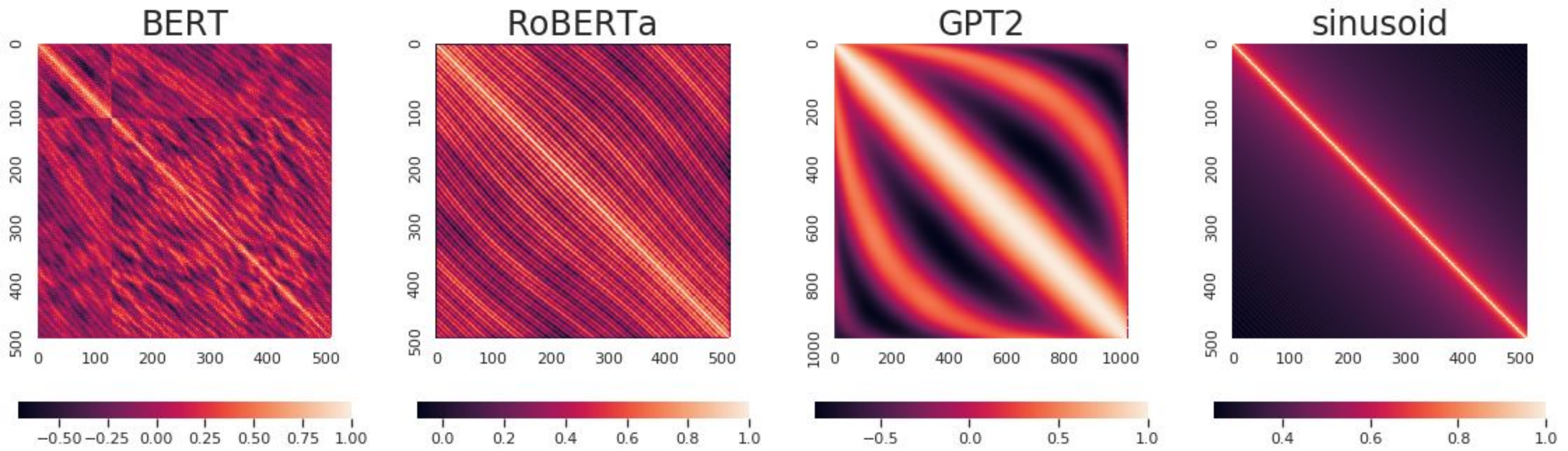
:

$$\boldsymbol{R}^d_{\Theta,m}\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{pmatrix} + \begin{pmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_{d-1} \\ x_d \end{pmatrix} \otimes \begin{pmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{pmatrix}$$

07 | Learned Positional Embeddings

Learned Positional Embedding

- Instead of crafting the positional embedding manually, we can treat those as any other parameter and **learn using SGD and backpropagation**.
- This has been implemented in BERT, RoBERTa, GPT2 etc. [Wang et. al. 2020]



Cosine similarity of positional embeddings at different positons

08 | References

References

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. **Attention is all you need.** *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.
- Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. **Self-Attention with Relative Position Representations.** *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.
- Yu-An Wang and Yun-Nung Chen. 2020. **What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding.** *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online. Association for Computational Linguistics.
- Su, Jianlin, Yu Lu, Shengfeng Pan, Bo Wen and Yunfeng Liu. 2021. **RoFormer: Enhanced Transformer with Rotary Position Embedding.** ArXiv abs/2104.09864