RoPE

Rotary Position Embedding

Papers

Attention is all you need:

- https://arxiv.org/pdf/1706.03762

RoFormer: Enhanced transformer with Rotary Position Embedding

- https://arxiv.org/pdf/2104.09864

Position Interpolation:

- https://arxiv.org/pdf/2306.15595

Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention

- https://arxiv.org/pdf/2006.16236

Videos

Efficient NLP: https://www.youtube.com/watch?v=o29P0Kpobz0

Code_your_own_ai: https://www.youtube.com/watch?v=GQPOtyITy54

DeepLearning Hero : https://www.youtube.com/watch?v=DvP8f7eWS7U

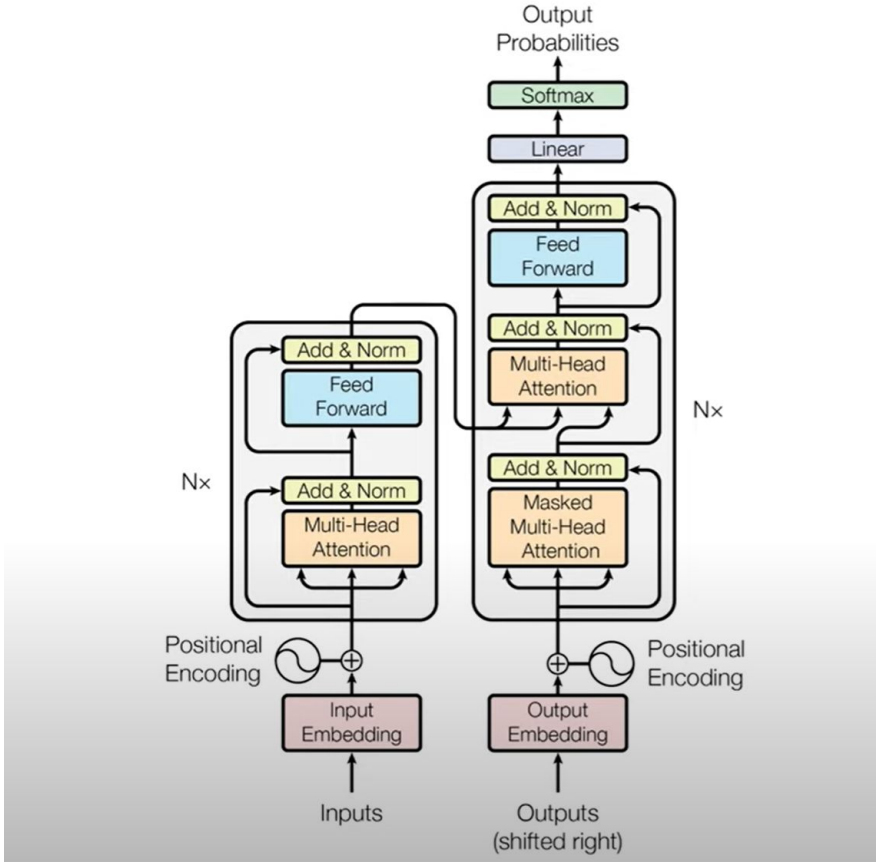
Web resources

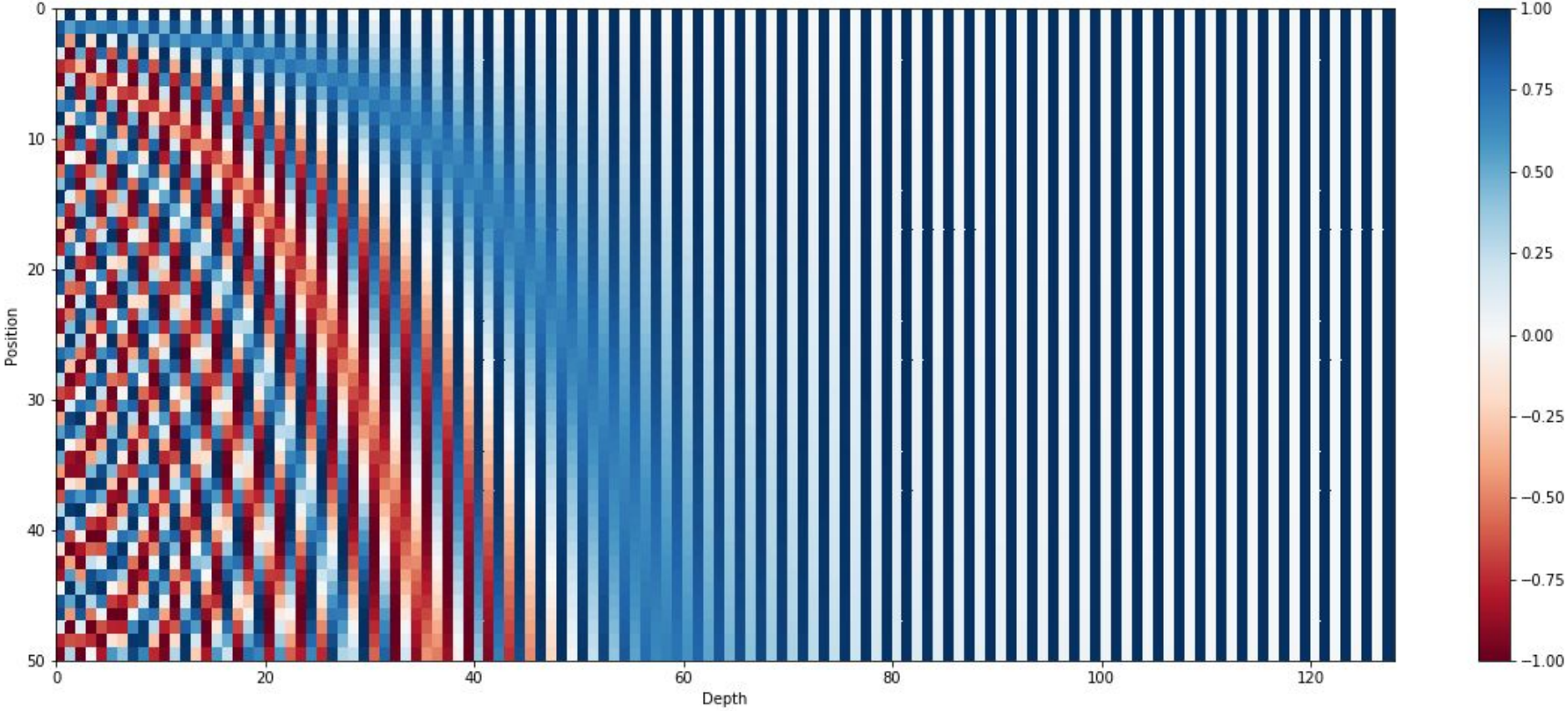
https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html

https://towardsdatascience.com/attention-is-all-you-need-e498378552f9

https://blog.eleuther.ai/rotary-embeddings/

Transformer




0
10
20
30
40
50
Position
0
20
40
60
80
100
120
Depth
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

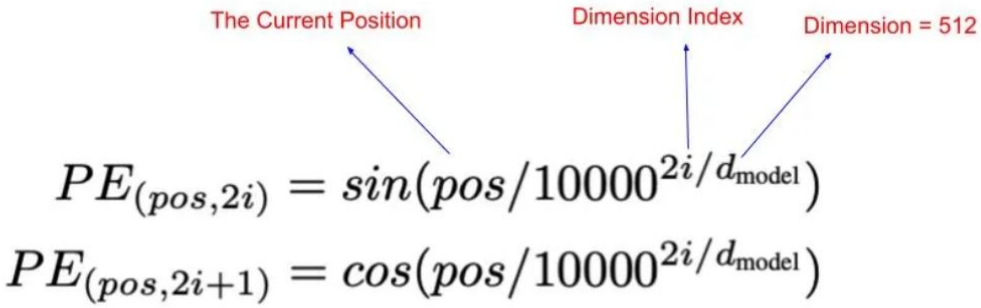


Figure 4: Positional Encodings Formula (source)

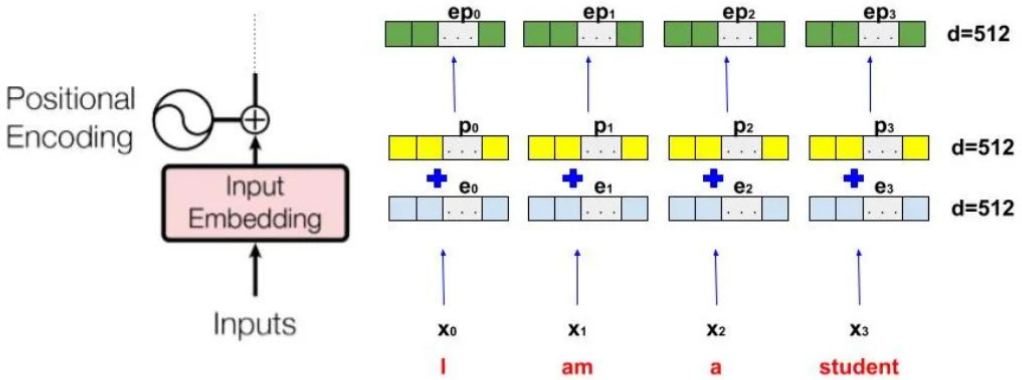


Figure 5: Adding Positional Encodings to the Embeddings to Generate Positional Embeddings (ep) (Image by Author)

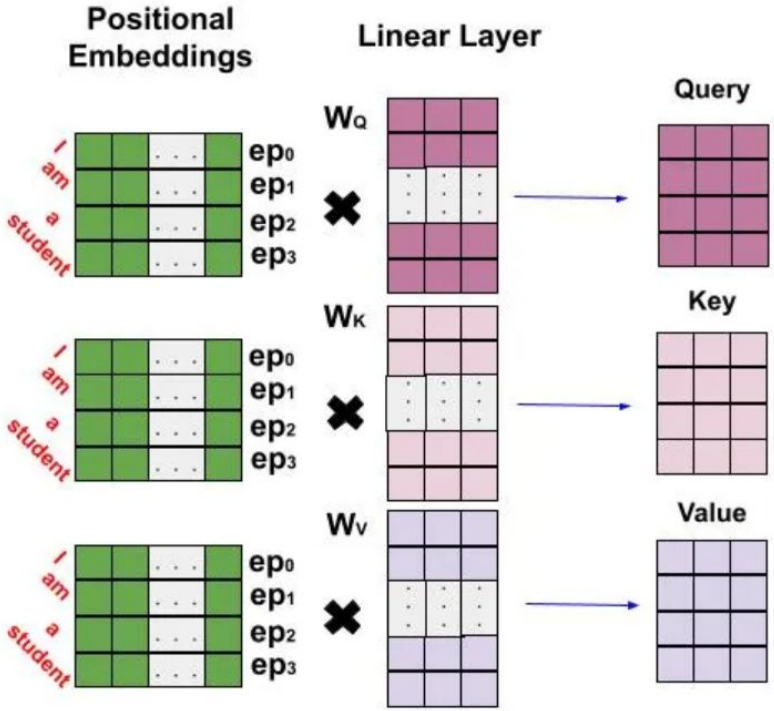
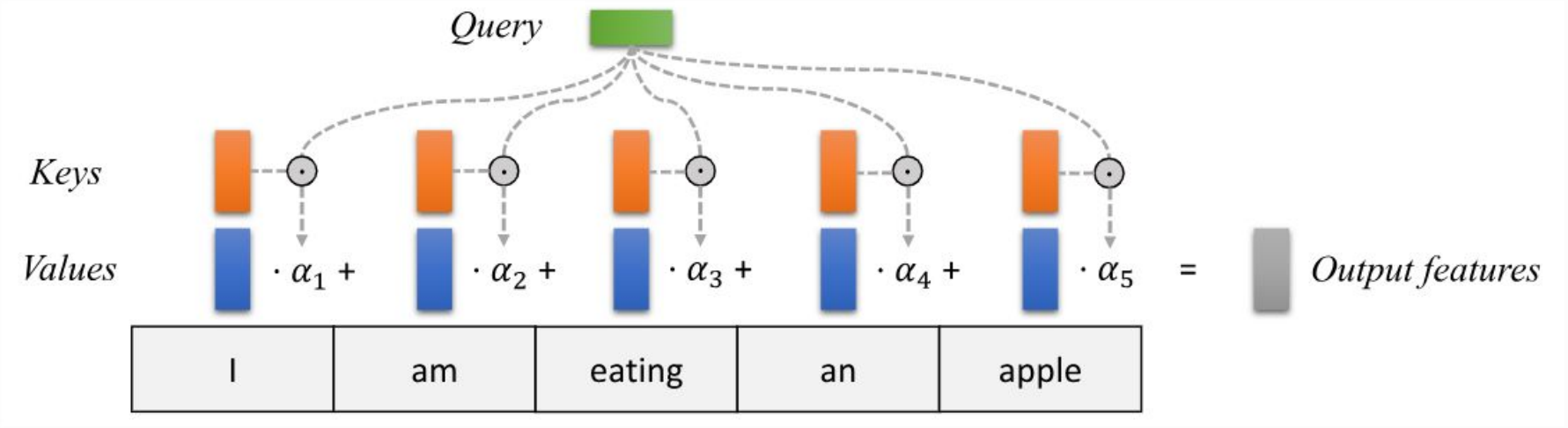


Figure 7: Generating the Query, Key and Value (Image by Author)

$$\alpha_i = \frac{\exp\left(f_{attn}\left(\text{key}_i, \text{query}\right)\right)}{\sum_j \exp\left(f_{attn}\left(\text{key}_j, \text{query}\right)\right)}, \quad \text{out} = \sum_i \alpha_i \cdot \text{value}_i$$

he attention over a sequence of words as follows:



Attention is all you need.

3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension d_{model} as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

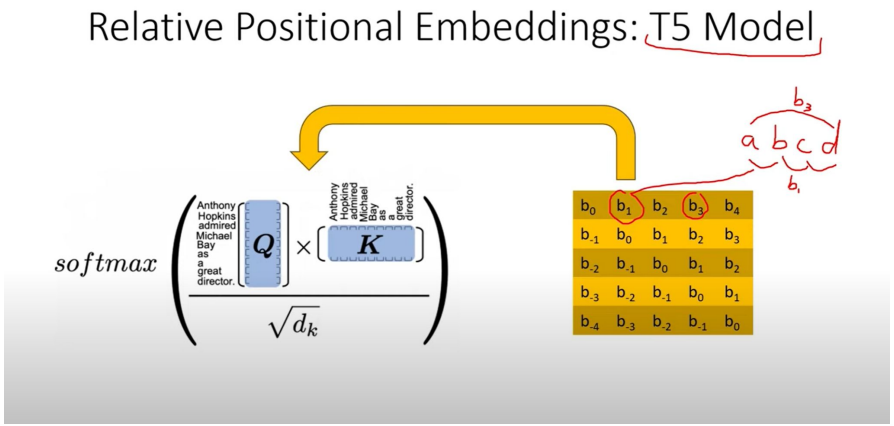
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

where pos is the position and i is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$. We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}.

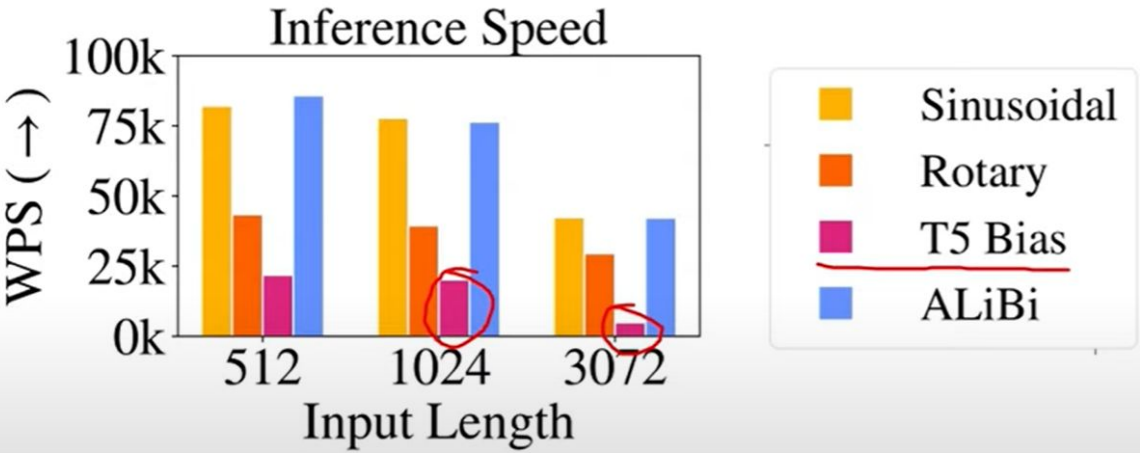
We also experimented with using learned positional embeddings [9] instead, and found that the two versions produced nearly identical results (see Table 3 row (E)). We chose the sinusoidal version because it may allow the model to extrapolate to sequence lengths longer than the ones encountered during training.

Relative position embedding

Efficient NLP



Relative embeddings are slow!



Why?

- Extra step in self-attention layer

Position embedding approaches

2.2 Absolute position embedding

A typical choice of Equation (1) is

$$f_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i, i) := \boldsymbol{W}_{t:t\in\{q,k,v\}}(\boldsymbol{x}_i + \boldsymbol{p}_i),$$

2.3 Relative position embedding

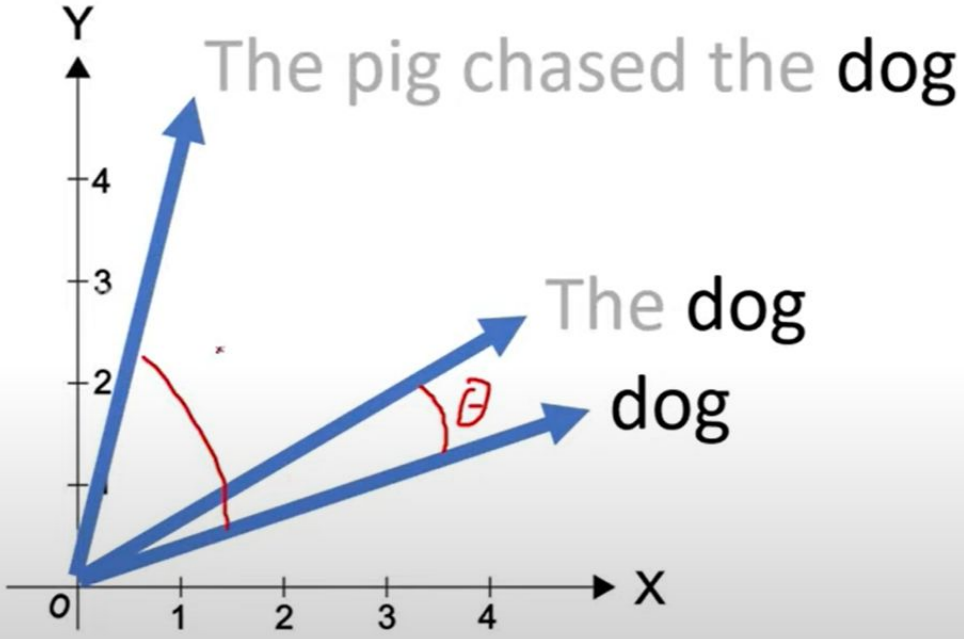
The authors of Shaw et al. [2018] applied different settings of Equation (1) as following:

$$f_q(\boldsymbol{x}_m) := \boldsymbol{W}_q\boldsymbol{x}_m$$
$$f_k(\boldsymbol{x}_n, n) := \boldsymbol{W}_k(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^k)$$
$$f_v(\boldsymbol{x}_n, n) := \boldsymbol{W}_v(\boldsymbol{x}_n + \tilde{\boldsymbol{p}}_r^v)$$

Rotary Position Embedding

$$
\begin{aligned}
f_q(\boldsymbol{x}_m, m) &= (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta} \\
f_k(\boldsymbol{x}_n, n) &= (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta} \\
g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) &= \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^* e^{i(m-n)\theta}]
\end{aligned}
$$

Rotary Positional Embeddings



$$
\begin{aligned}
\boldsymbol{q}_m &= f_q(\boldsymbol{x}_m, m) \\
\boldsymbol{k}_n &= f_k(\boldsymbol{x}_n, n) \\
\boldsymbol{v}_n &= f_v(\boldsymbol{x}_n, n),
\end{aligned}
$$

$$
a_{m,n} = \frac{\exp(\frac{\boldsymbol{q}_m^\intercal \boldsymbol{k}_n}{\sqrt{d}})}{\sum_{j=1}^{N} \exp(\frac{\boldsymbol{q}_m^\intercal \boldsymbol{k}_j}{\sqrt{d}})}
$$

$$
\mathbf{o}_m = \sum_{n=1}^{N} a_{m,n} \boldsymbol{v}_n
$$