Topics in NLP (CS6803)

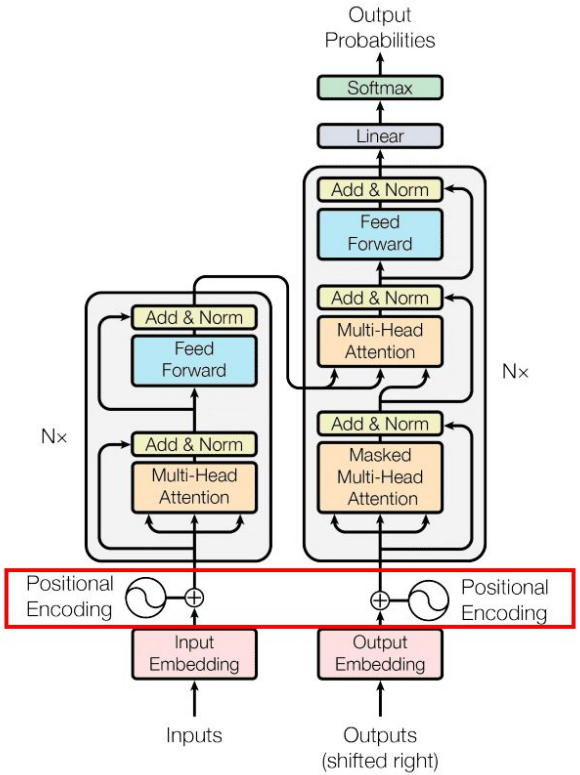
Remember where you are: Positional Embeddings

Dr. Maunendra Sankar Desarkar

Positional Embeddings

- Why do we need Positional Embeddings?
- Where are they introduced in the architecture/computation?

- PE = f(pos), where pos is the position
- Many choices exist



Positional embeddings in Vanilla Transformers

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

Query, key and value as functions of base embedding and position

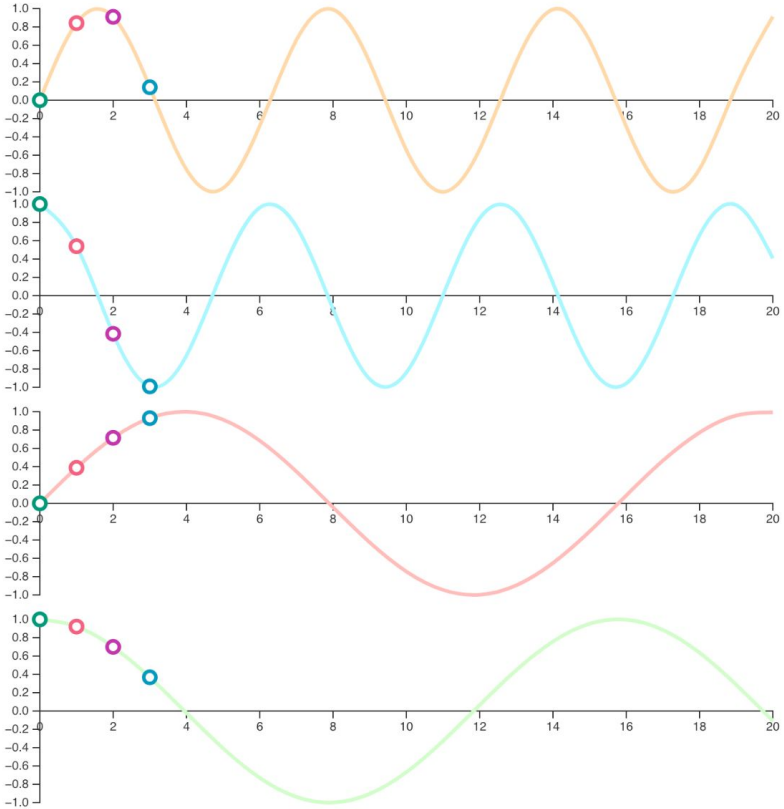
Additive and modeled using trigonometric functions

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

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

$$\text{PE}(1) = [\sin(\tfrac{1}{10000^{2\times0/4}}), \cos(\tfrac{1}{10000^{2\times0/4}}), \sin(\tfrac{1}{10000^{2\times1/4}}), \cos(\tfrac{1}{10000^{2\times1/4}})]$$
$$\text{PE}(2) = [\sin(\tfrac{2}{10000^{2\times0/4}}), \cos(\tfrac{2}{10000^{2\times0/4}}), \sin(\tfrac{2}{10000^{2\times1/4}}), \cos(\tfrac{2}{10000^{2\times1/4}})]$$
$$\text{PE}(3) = [\sin(\tfrac{3}{10000^{2\times0/4}}), \cos(\tfrac{3}{10000^{2\times0/4}}), \sin(\tfrac{3}{10000^{2\times1/4}}), \cos(\tfrac{3}{10000^{2\times1/4}})]$$
$$\text{PE}(4) = [\sin(\tfrac{4}{10000^{2\times0/4}}), \cos(\tfrac{4}{10000^{2\times0/4}}), \sin(\tfrac{4}{10000^{2\times1/4}}), \cos(\tfrac{4}{10000^{2\times1/4}})]$$
$$\text{PE}(5) = [\sin(\tfrac{5}{10000^{2\times0/4}}), \cos(\tfrac{5}{10000^{2\times0/4}}), \sin(\tfrac{5}{10000^{2\times1/4}}), \cos(\tfrac{5}{10000^{2\times1/4}})]$$
$$\text{PE}(6) = [\sin(\tfrac{6}{10000^{2\times0/4}}), \cos(\tfrac{6}{10000^{2\times0/4}}), \sin(\tfrac{6}{10000^{2\times1/4}}), \cos(\tfrac{6}{10000^{2\times1/4}})]$$

Visualizing the values across dimensions



p0	p1	p2	p3	
0.000	0.841	0.909	0.141	**i=0**
1.000	0.540	-0.416	-0.990	**i=1**
0.000	0.388	0.715	0.930	**i=2**
1.000	0.922	0.699	0.368	**i=3**

Positional Encoding

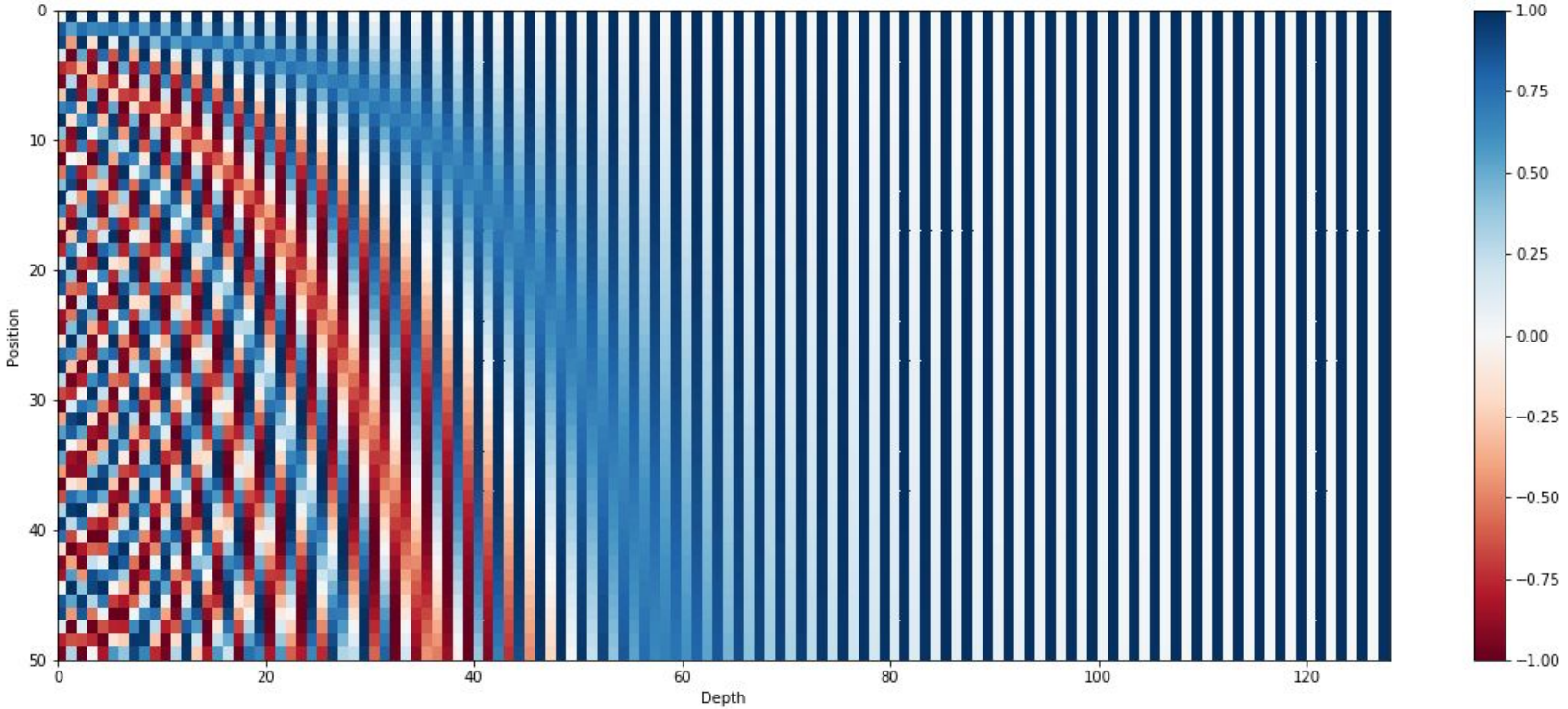
$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{2i/d_{\text{model}}}})$$

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

Settings: d = 20

Even if the dimension (*d*) has changed, values for **i=0** and **i=1** haven't. That's because **PE** formula has a divisor equal to **1** in both cases, despite the value of *d*.

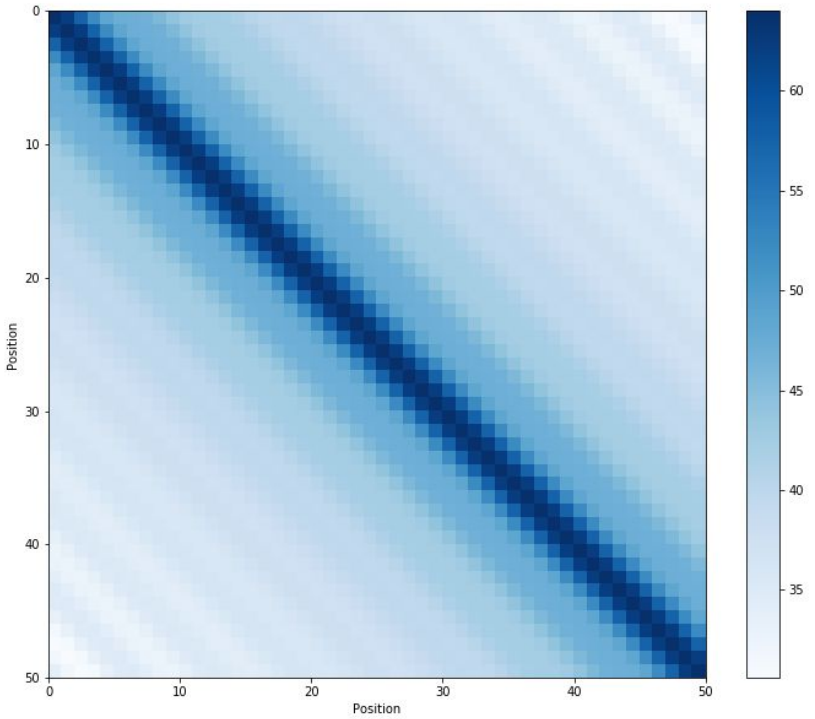
Visualizing the values across dimensions



Properties of the scheme

- Deterministic in nature.
  - A unique encoding for each position/time-step
- Generalizes to longer sentences easily
- The values in the embedding vector are bounded
- Distance between neighboring positions
  - Symmetrical
  - Decays with time
- For any fixed offset k, PE(i+k)= lin(PE(i))

$$M.\begin{bmatrix}\sin(\omega_k \cdot t)\\ \cos(\omega_k \cdot t)\end{bmatrix} = \begin{bmatrix}\sin(\omega_k \cdot (t+\phi))\\ \cos(\omega_k \cdot (t+\phi))\end{bmatrix}$$



Embeddings at fixed offsets

Let M be a 2×2 matrix, we want to find u_1, v_1, u_2 and v_2 so that:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} . \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . (t + \phi)) \\ \cos(\omega_k . (t + \phi)) \end{bmatrix}$$

By applying the addition theorem, we can expand the right hand side as follows:

$$\begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix} . \begin{bmatrix} \sin(\omega_k . t) \\ \cos(\omega_k . t) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k . t) \cos(\omega_k . \phi) + \cos(\omega_k . t) \sin(\omega_k . \phi) \\ \cos(\omega_k . t) \cos(\omega_k . \phi) - \sin(\omega_k . t) \sin(\omega_k . \phi) \end{bmatrix}$$

Which result in the following two equations:

$$\begin{aligned} u_1 \sin(\omega_k . t) + v_1 \cos(\omega_k . t) &= \cos(\omega_k . \phi) \sin(\omega_k . t) + \sin(\omega_k . \phi) \cos(\omega_k . t) && (1) \\ u_2 \sin(\omega_k . t) + v_2 \cos(\omega_k . t) &= -\sin(\omega_k . \phi) \sin(\omega_k . t) + \cos(\omega_k . \phi) \cos(\omega_k . t) && (2) \end{aligned}$$

By solving above equations, we get:

$$\begin{aligned} u_1 &= \cos(\omega_k . \phi) & v_1 &= \sin(\omega_k . \phi) \\ u_2 &= -\sin(\omega_k . \phi) & v_2 &= \cos(\omega_k . \phi) \end{aligned}$$

So the final transformation matrix M is:

$$M_{\phi,k} = \begin{bmatrix} \cos(\omega_k . \phi) & \sin(\omega_k . \phi) \\ -\sin(\omega_k . \phi) & \cos(\omega_k . \phi) \end{bmatrix}$$

References

- Attention Is All You Need
- Positional embedding visualization
- Positional Encoding
- Transformer Architecture: The Positional Encoding

Relative Positional Embeddings

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

Query, key and value as functions of based embedding and position

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

p_r s are relative position embeddings

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n,$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n} + \mathbf{u}^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \mathbf{v}^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n}$$

Relative Positional Embeddings

- Considers relative positions for token-pairs
- If there are T tokens, T x 2*(T-1) dimensional pairwise embedding matrix is created
  - Dimension 1: Position of the current word of interest
  - Dimension 2: Positional distance from the current word

- As opposed to adding the positional information element-wise directly to the token embeddings, **relative positional information is added to keys and values during attention calculation.**

- Kicks in during the attention computation

https://jaketae.github.io/study/relative-positional-encoding/

https://arxiv.org/pdf/1803.02155

Self-Attention with Relative Position Representations

Self-attention

- Input: $x=\{x_1, x_2, \ldots\}$
- Output: $z=\{z_1, z_2, \ldots\}$

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

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

Relation-aware Self-attention

- Encodes pairwise information between positions i and j
- Maintains two vectors per (i,j) pair: a_{ij}^V, a_{ij}^K

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

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

Learns the values of a_{ij}^V, a_{ij}^K
Clips the distance i-j after a certain point

Shaw, P., Uszkoreit, J. and Vaswani, A., 2018, June. Self-Attention with Relative Position Representations. NAACL 2018

Relative Positional Embeddings

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

Query, key and value as functions of based embedding and position

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

p_r s are relative position embeddings

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n,$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n} + \mathbf{u}^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \mathbf{v}^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n}$$

Some other variations: Transformer-XL

Processes text that goes beyond the allowed sequence length by splitting the text into multiple segments. Uses relative positional encoding. Why?

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n,$$

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n} + \mathbf{u}^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \mathbf{v}^\intercal \boldsymbol{W}_q^\intercal \widetilde{\boldsymbol{W}}_k \tilde{\boldsymbol{p}}_{m-n}$$

- Replace all appearances of the absolute positional embedding for computing key vectors with its relative counterpart … essentially reflects the prior that only the relative distance matters for where to attend.
- Since the query vector is the same for all query positions, it suggests that the attentive bias towards different words should remain the same regardless of the query position
- Deliberately separate the two weight matrices for producing the content-based key vectors and location-based key vectors

Dai Z, Yang Z, Yang Y, Carbonell JG, Le Q, Salakhutdinov R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. ACL 2019

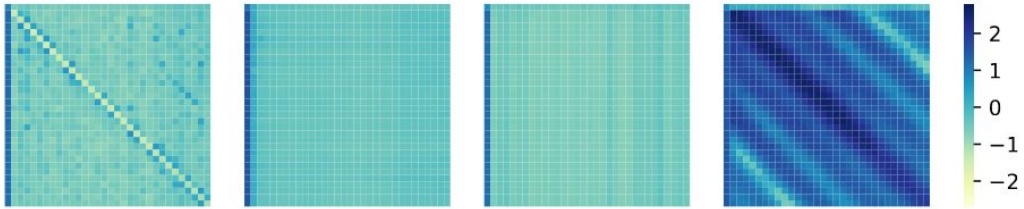
Some other variations

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + \boldsymbol{p}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{p}_n,$$

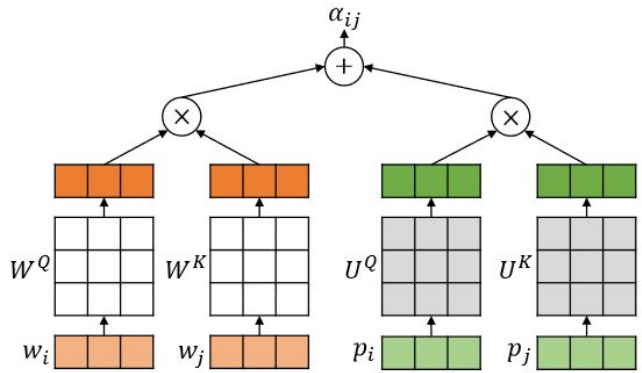
$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = \boldsymbol{x}_m^\intercal \boldsymbol{W}_q^\intercal \boldsymbol{W}_k \boldsymbol{x}_n + b_{i,j}$$

From T5 Paper: We use a simplified form of position embeddings where each "embedding" is simply a scalar that is added to the corresponding logit used for computing the attention weights. For efficiency, we also share the position embedding parameters across all layers in our model, though within a given layer each attention head uses a different learned position embedding.

Some other variations



From left to right: word-to-word, word-to-position, position-to-word, and position-to-position correlation matrices. In each matrix, the (i-th, j-th) element is the correlation between i-th word/position and j-th word/position.



$$q_m^\intercal k_n = x_m^\intercal W_q^\intercal W_k x_n + p_m^\intercal U_q^\intercal U_k p_n + b_{i,j}$$

we propose a new positional encoding method called Transformer with Untied Positional Encoding (TUPE). In the self-attention module, TUPE computes the word contextual correlation and positional correlation separately with different parameterizations and then adds them together. This design removes the mixed and noisy correlations over heterogeneous embeddings and offers more expressiveness by using different projection matrices.

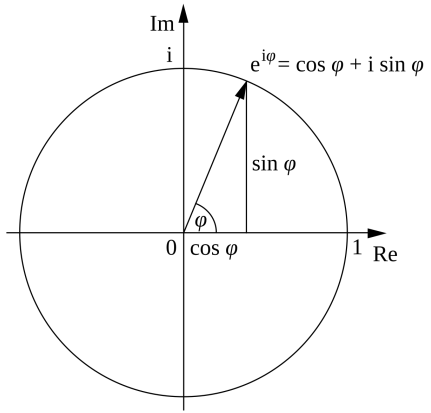
Ke, G., He, D. and Liu, T.Y., Rethinking Positional Encoding in Language Pre-training. In *ICLR 2021*

Rotary Position Embeddings

In order to incorporate relative position information, we require the inner product of query q_m and key k_n to be formulated by a function g, which takes only the word embeddings x_m, x_n, and their relative position $m - n$ as input variables. In other words, we hope that the inner product encodes position information only in the relative form

$$\langle f_q(\boldsymbol{x}_m, m), f_k(\boldsymbol{x}_n, n)\rangle = g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n).$$

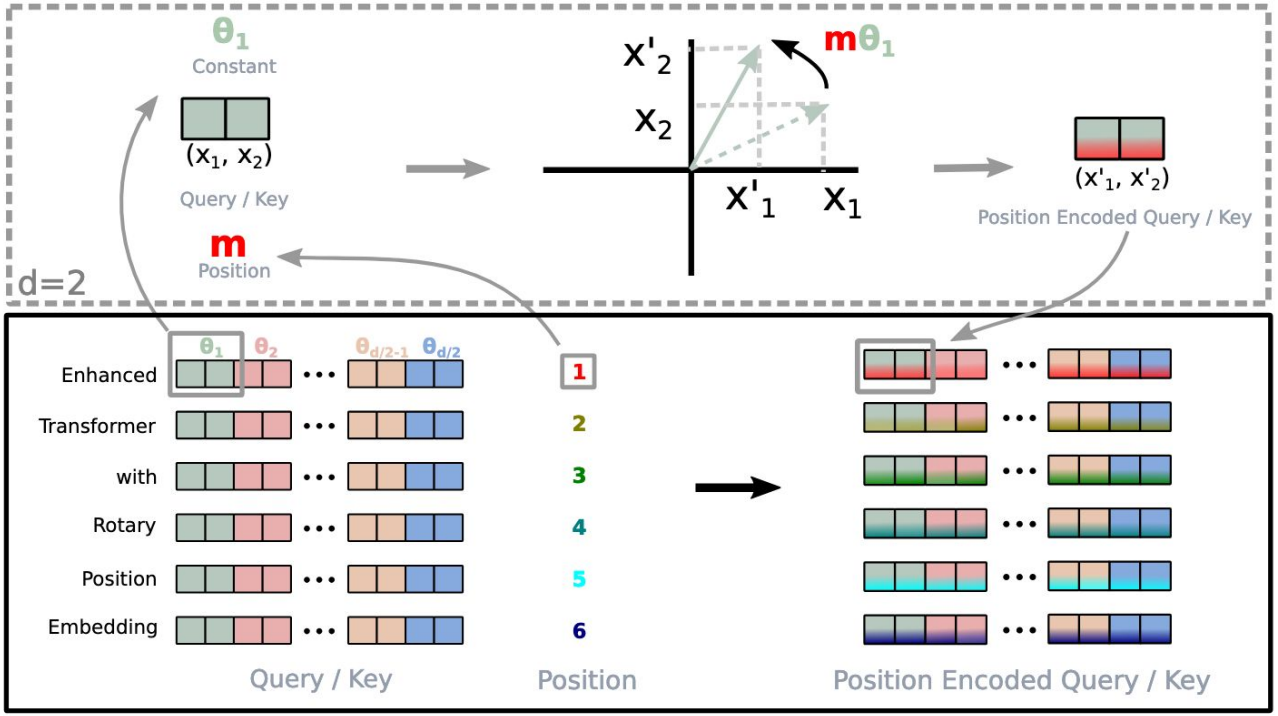
$$\begin{aligned} f_q(\boldsymbol{x}_m, m) &= (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta} \\ f_k(\boldsymbol{x}_n, n) &= (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta} \\ g(\boldsymbol{x}_m, \boldsymbol{x}_n, m-n) &= \mathrm{Re}[(\boldsymbol{W}_q \boldsymbol{x}_m)(\boldsymbol{W}_k \boldsymbol{x}_n)^*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



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

Rotary Position Embeddings

- Break d dimensional embeddings into d/2 vectors of length 2: [(x1, x2), (x3, x4), ..,]
- Rotate each two dimensional vector by an angle mθ
- m is the position-difference between the query and key tokens



Rotary Position Embeddings

$$\theta_i = 10000^{-2i/d}.$$

- Setting θ like this provides a long-term decay property - the inner-product will decay when the relative position increase.
- This property coincides with the intuition that a pair of tokens with a long relative distance should have less connection

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

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

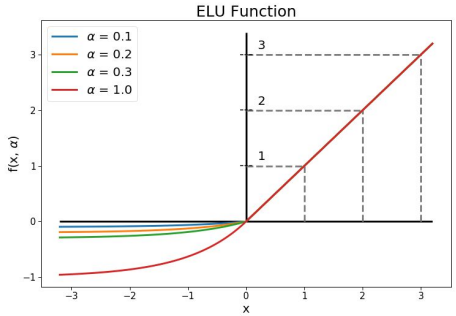
$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_q\boldsymbol{x}_m)^\intercal(\boldsymbol{R}^d_{\Theta,n}\boldsymbol{W}_k\boldsymbol{x}_n) = \boldsymbol{x}^\intercal\boldsymbol{W}_q R^d_{\Theta,n-m}\boldsymbol{W}_k\boldsymbol{x}_n$$

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V})_m = \frac{\sum_{n=1}^N \left(\boldsymbol{R}^d_{\Theta,m}\phi(\boldsymbol{q}_m)\right)^\intercal\left(\boldsymbol{R}^d_{\Theta,n}\varphi(\boldsymbol{k}_n)\right)\boldsymbol{v}_n}{\sum_{n=1}^N \phi(\boldsymbol{q}_m)^\intercal\varphi(\boldsymbol{k}_n)}.$$

Rotary Position Embeddings

$$\phi(x) = \varphi(x) = \mathrm{elu}(x) + 1$$

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{if } x \leq 0 \end{cases}$$



$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_m = \frac{\sum_{n=1}^{N} \left(\boldsymbol{R}^d_{\Theta,m}\phi(\boldsymbol{q}_m)\right)^\intercal \left(\boldsymbol{R}^d_{\Theta,n}\varphi(\boldsymbol{k}_n)\right)\boldsymbol{v}_n}{\sum_{n=1}^{N} \phi(\boldsymbol{q}_m)^\intercal \varphi(\boldsymbol{k}_n)}.$$

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

Summary: How RoPE works?

- First computes the query ($W_q * x_m$) and key ($W_k * x_n$) vectors for each word (or token) in the sequence.
- Multiply the query and key vectors by a rotation matrix. This matrix is determined by the absolute positions of the tokens (e.g., their positions in the sentence).
- Compute the inner product between the rotated query and key vectors.
- The result of this inner product is an attention matrix that depends on the relative positions of the tokens, allowing the model to capture positional relationships more effectively.

Other methods

CAPE: Encoding Relative Positions with Continuous Augmented Positional Embeddings