RoFormer

Enhanced Transformer with Rotary Position Embedding

Su, Lu, Pan, Murtadha, Wen & Liu · Zhuiyi Technology · arXiv:2104.09864v5 (2023) . Cited by 5998

Position Encoding

Rotary Matrix

Integrated into HuggingFace Transformers

Presentation Overview

01
Research Question & Motivation
Why does position encoding matter? What is missing?

02
Background & Related Work
Absolute vs. relative position encoding

03
Proposed Method: RoPE
Formulation, derivation, and the rotation matrix

05
Experiments & Results
Machine translation, pre-training, GLUE benchmarks

06
Conclusions & Limitations
Key takeaways and open questions

Research Question & Motivation

Research Question

How can we encode absolute position information into transformer self-attention such that the model naturally captures relative positional dependencies — without modifying the additive embedding framework or sacrificing linear-attention compatibility?

Self-Attention is Position-Agnostic

The dot-product attention mechanism (Vaswani et al., 2017) has no inherent notion of token order. Without explicit position signals, the model treats sequences as sets.

Relative Position is Semantically Meaningful

Nearby tokens interact more strongly than distant ones. A desired property: attention scores should decay as relative distance $|m - n|$ increases.

Additive Encoding Limitations

Common approaches add position vectors p_i to token embeddings before projection. This complicates relative-position computation and is incompatible with linear attention.

No Unified Multiplicative Framework

Prior relative PE methods (Shaw et al., Dai et al., He et al.) are additive decompositions — none encodes relative position purely via rotation of representations.

Background: Position Encoding in Transformers

Absolute Position Embedding

General form (Vaswani et al., 2017; Devlin et al., 2019):

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

Sinusoidal encoding (fixed):

$$\begin{cases} \boldsymbol{p}_{i,2t} &= \sin(k/10000^{2t/d}) \\ \boldsymbol{p}_{i,2t+1} &= \cos(k/10000^{2t/d}) \end{cases}$$

Trainable vectors (BERT, GPT, ALBERT)

Limitation: position added to context — not naturally relative; long-range interaction unclear.

Related Work →

Relative Position Embedding

Shaw et al. (2018) — add $\tilde{p}_r$ to keys/values:

$$q_m^\intercal k_n = x_m^\intercal W_q^\intercal W_k x_n + x_m^\intercal W_q^\intercal W_k p_n + p_m^\intercal W_q^\intercal W_k x_n + p_m^\intercal W_q^\intercal W_k p_n,$$

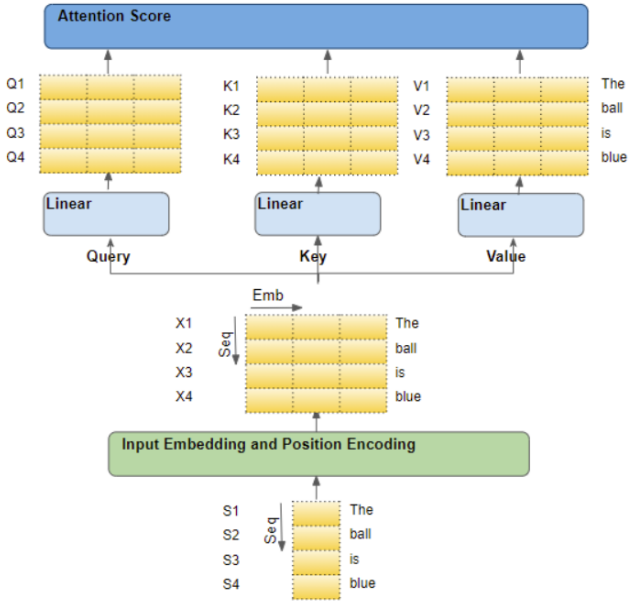
Transformer-XL (Dai et al., 2019):

$$q_m^\intercal k_n = x_m^\intercal W_q^\intercal W_k x_n + x_m^\intercal W_q^\intercal \widetilde{W}_k \tilde{p}_{m-n} + \mathrm{u}^\intercal W_q^\intercal W_k x_n + \mathrm{v}^\intercal W_q^\intercal \widetilde{W}_k \tilde{p}_{m-n}$$

DeBERTa, T5, TUPE follow similar additive decompositions of Eq. $q_m^T k_n$.

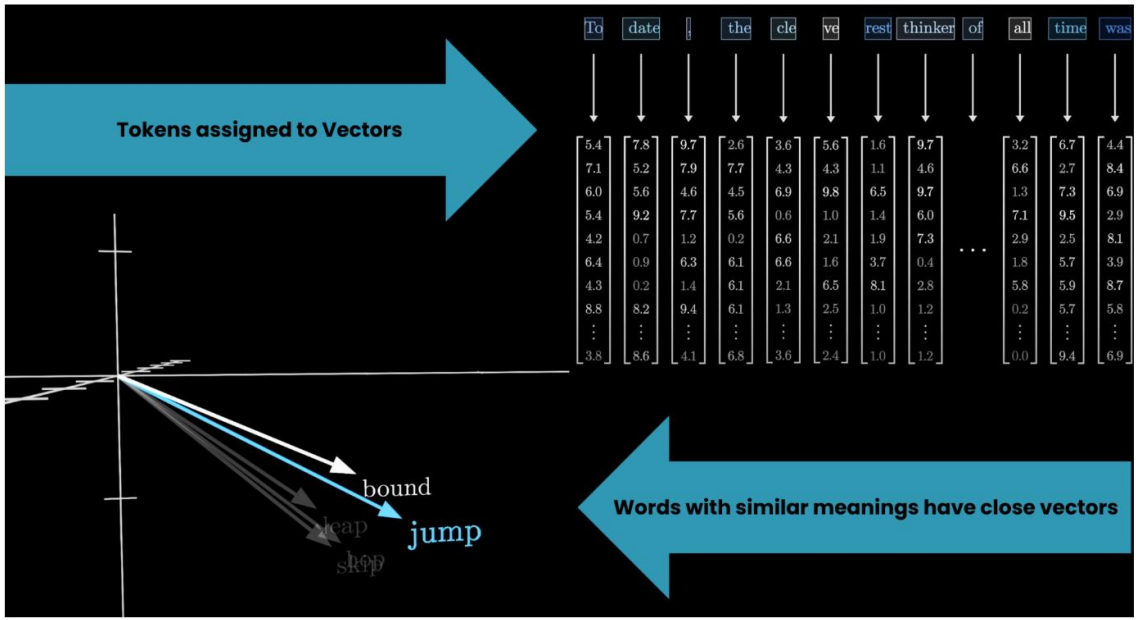
Key insight missing: encode relative position via rotation, not addition.

Background: Transformers



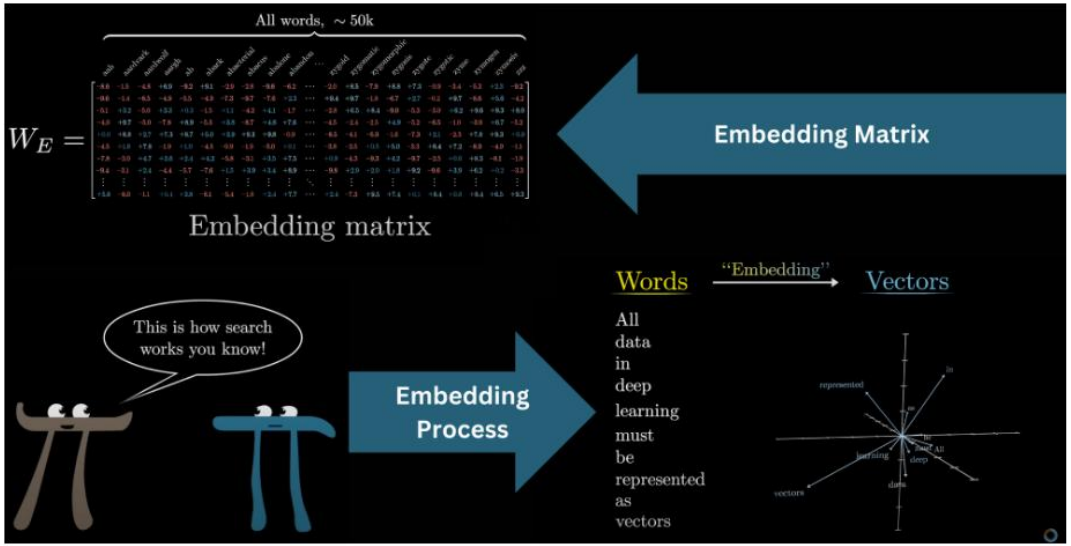
Background: Embedding in Transformers

https://www.3blue1brown.com/lessons/gpt/



In time-series data, we can treat each sample point as a single word, and the value of the sample point is the vector.
Or we can use patching, and each patch is a vector.

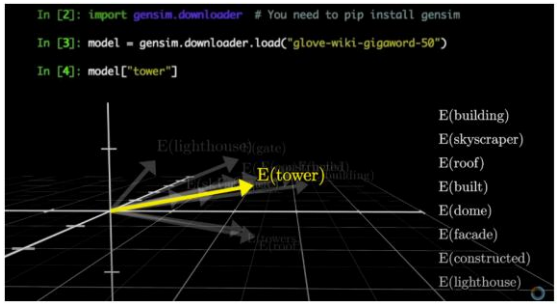
Background: Embedding in Transformers



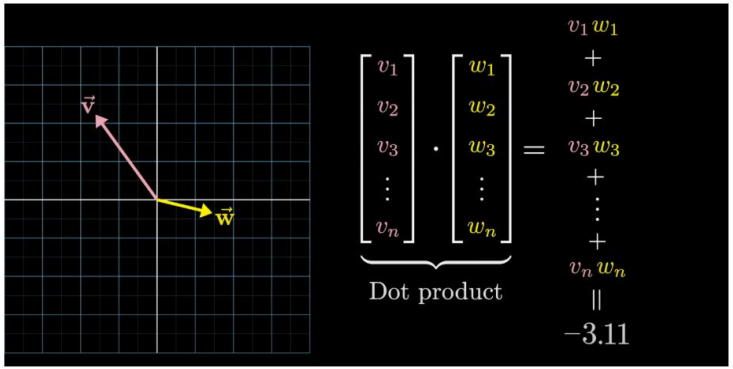
Turning words into vectors is often called *embedding the word*, which invites thinking of these vectors geometrically as points or directions in some space. Visualizing word embeddings tends to be very high-dimensional. For GPT-3, they have 12,288 dimensions.

Background: Embedding in Transformers

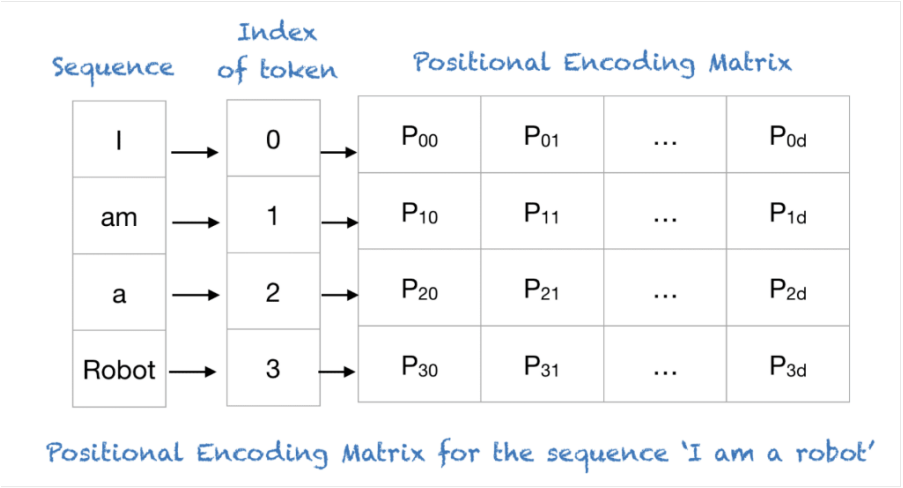
A simple word-to-vector model is running, and when we run a search for all words whose embeddings are closest to that of *tower*, they all generally have the same vibe.

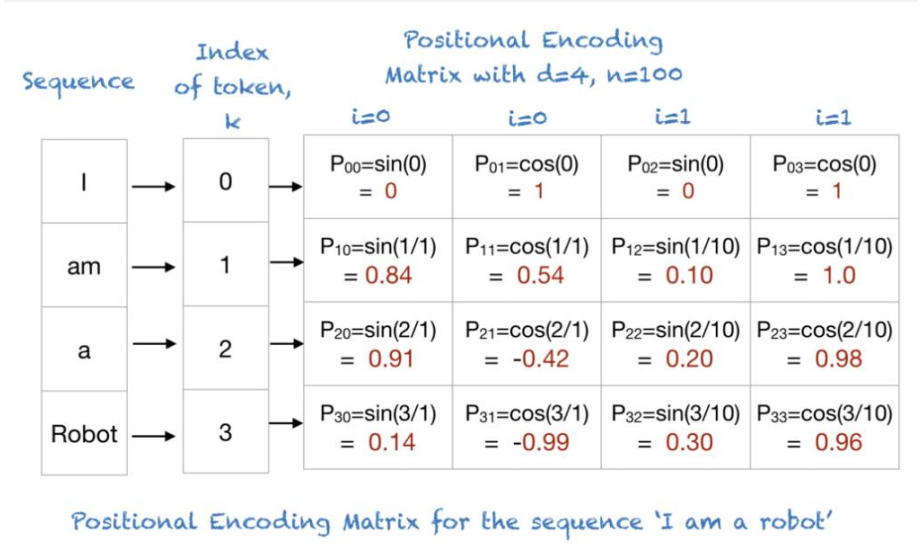


Mathematical intuition: The dot product of two vectors measures how well they align. Computationally, dot products involve multiplying all aligned components and adding the result. Geometrically, the dot product is positive when the vectors point in a similar direction, zero if they're *perpendicular*, and negative when they point in opposite directions.



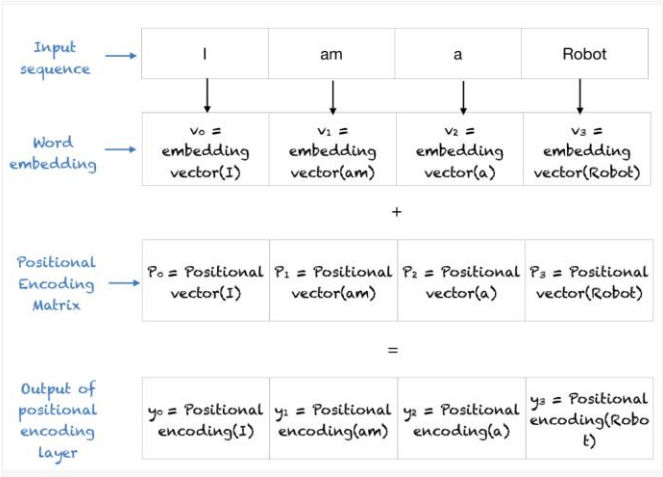
Background: Position Encoding in Transformers

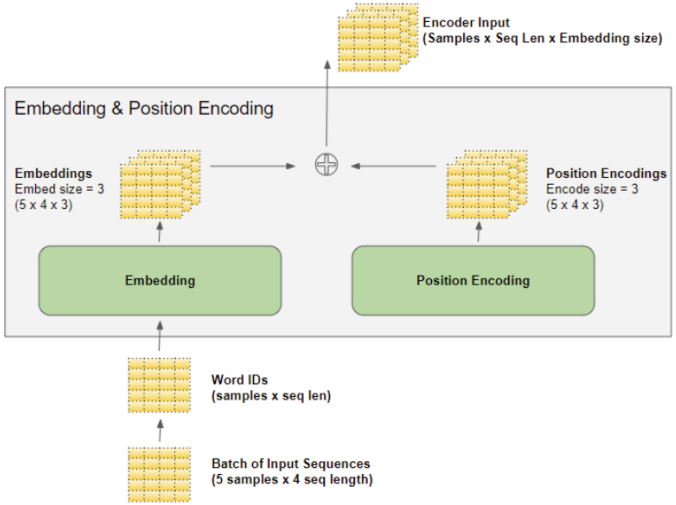




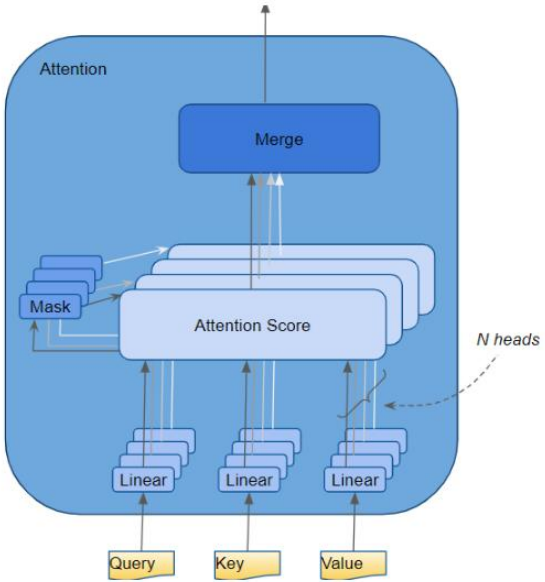
https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

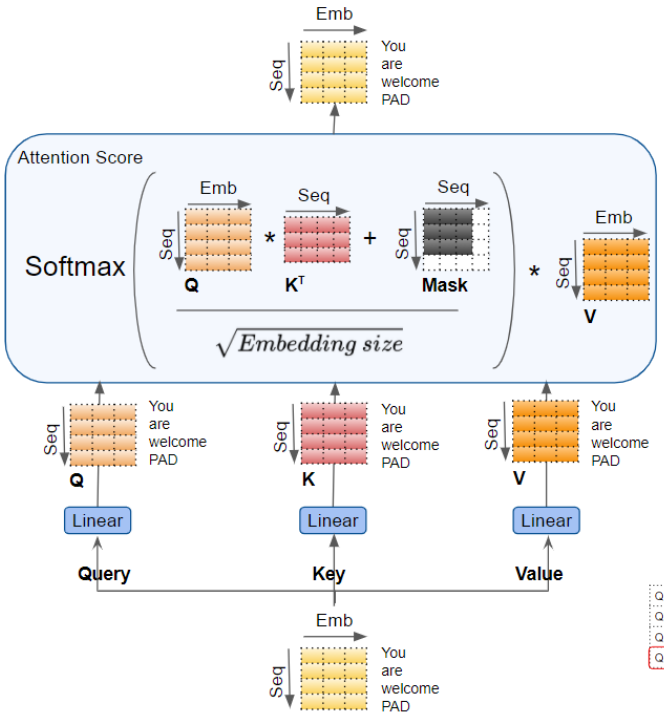
Background: Embed & Position Encoding in Transformers



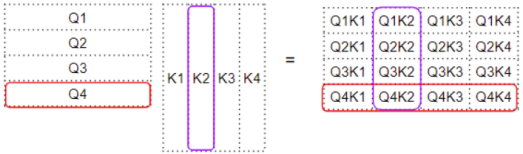


Background: Attention Score in Transformers

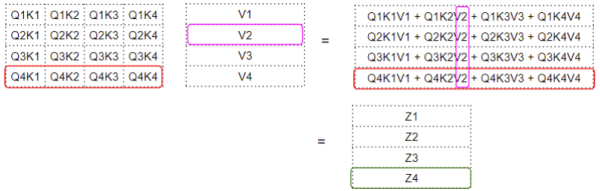




The column in the fourth row corresponds to a dot product between the fourth Query word and every Keyword.



matrix multiply between this intermediate 'factor' matrix and the Value (V) matrix, to produce the attention score that is output by the attention module.

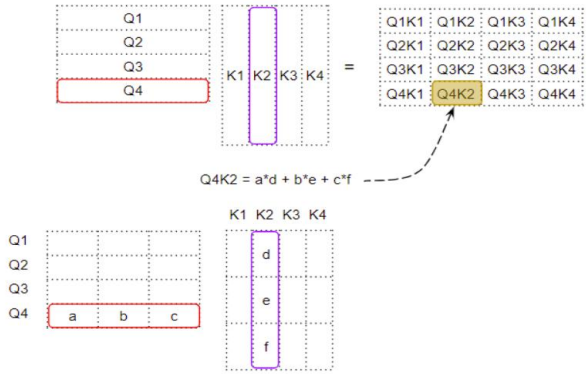


https://towardsdatascience.com/transformers-explained-visually-part-2-how-it-works-step-by-step-b49fa4a64f34/

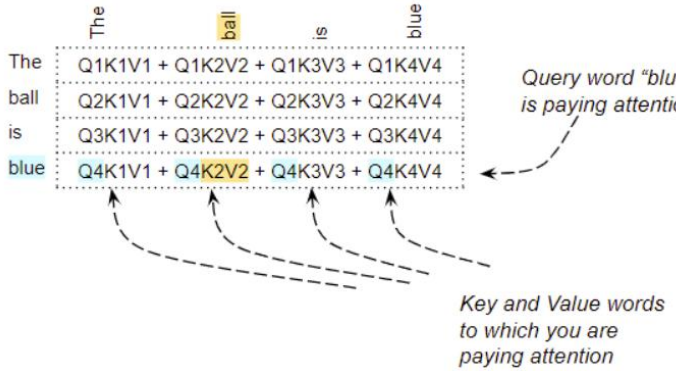
https://towardsdatascience.com/transformers-explained-visually-not-just-how-but-why-they-work-so-well-d840bd61a9d3/

Background: Attention Score in Transformers

Query, Key, and Value rows are actually vectors with an Embedding dimension.



The Query word can be interpreted as the word *for which* we are calculating Attention. The Key and Value word is the word *to which* we are paying attention ie. how relevant is that word to the Query word.

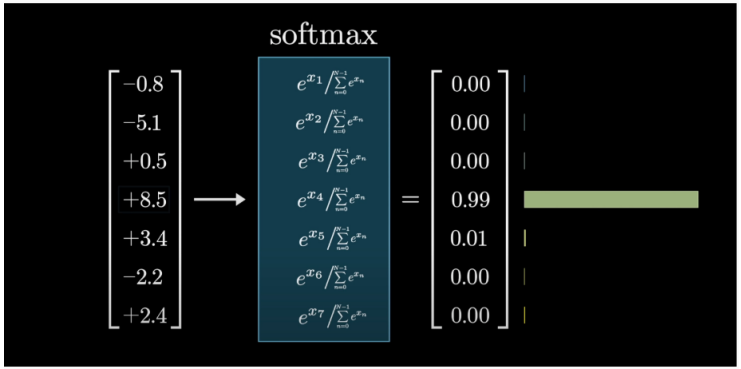


When we do a dot product between two vectors, we multiply pairs of numbers and then sum them up.
•If the two paired numbers (eg. 'a' and 'd' above) are both positive or both negative, then the product will be positive. The product will increase the final summation.
•If one number is positive and the other negative, then the product will be negative. The product will reduce the final summation.
•If the product is positive, the larger the two numbers, the more they contribute to the final summation.
This means that if the signs of the corresponding numbers in the two vectors are aligned, the final sum will be

Background: Visual of a Transformers

https://poloclub.github.io/transformer-explainer/

Softmax turns an arbitrary list of numbers into a valid distribution, in such a way that the largest values end up closest to 1, and the smaller values end up closer to 0.



Proposed Method: RoPE — Core Formulation

Core Idea

Require the inner product $\langle f_q(x_m, m), f_k(x_n, n)\rangle$ to depend only on x_m, x_n, and the relative position $(m - n)$ — achieved by applying a position-dependent rotation matrix to query and key vectors.

Eq. (11) — Relative Dependency Constraint

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

The attention score must be a function of the relative position only.

Eq. (12) — 2D Solution (complex number form)

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

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

Multiplying by $e^{im\theta}$ rotates the vector by angle $m\theta$ in the complex plane.

Eq. (14 / 16) — General d-dimensional Form

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

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

The d-dim space is split into d/2 2D sub-spaces. R^d_{Θ} is a sparse orthogonal rotation matrix.

Properties of RoPE: The Rotation Matrix & 2D Derivation

Rotary Matrix $R^d_{\Theta,m}$ (Eq. 15)

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

$$\theta_i = 10000^{-2(i-1)/d}, \qquad i \in \{1, \ldots, d/2\}$$

R^d_Θ is orthogonal → $\|R^d_{\Theta,m}\, x\| = \|x\|$ (norm-preserving)

Derivation Sketch (2D Case)

1 — Express in polar form

Decompose f_q , f_k , g into radial R(·) and angular Θ(·) components in ℂ. Plug into the constraint $\langle f_q , f_k \rangle = g(\cdot, \cdot, m-n)$.

2 — Radial components are position-free

Setting m = n shows $R_q(x, m) = \|q\|$ and $R_k(x, n) = \|k\|$ — independent of position.

$$R_q(\boldsymbol{x}_q, m) = R_q(\boldsymbol{x}_q, 0) = \|\boldsymbol{q}\|$$
$$R_k(\boldsymbol{x}_k, n) = R_k(\boldsymbol{x}_k, 0) = \|\boldsymbol{k}\|$$
$$R_g(\boldsymbol{x}_q, \boldsymbol{x}_k, n-m) = R_g(\boldsymbol{x}_q, \boldsymbol{x}_k, 0) = \|\boldsymbol{q}\|\|\boldsymbol{k}\|$$

3 — Angular component is an arithmetic progression

$\Theta_f(x, m) = \varphi(m) + \theta_{q/k}$, where ϕ(m) = mθ + γ — a linear function of position (Eq. 30).

4 — Final solution (simply set γ = 0)

(Eq. 33)

$$f_q(\boldsymbol{x}_m, m) = (\boldsymbol{W}_q \boldsymbol{x}_m) e^{im\theta},$$
$$f_k(\boldsymbol{x}_n, n) = (\boldsymbol{W}_k \boldsymbol{x}_n) e^{in\theta}.$$

RoPE in Action: Implementation Diagram

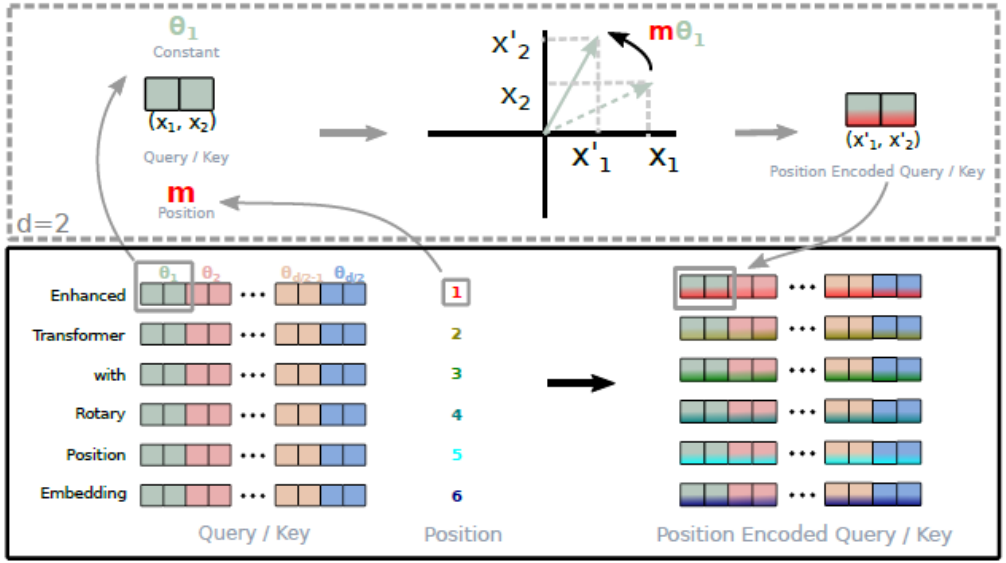


Figure 1: Implementation of Rotary Position Embedding(RoPE).

Figure 1 (Su et al., 2023): Each token's Query/Key vector (left) is multiplied by the position-indexed rotation matrix R^d_{Θ,m}, yielding a position-encoded Query/Key (right). The rotation angle is proportional to the token position m.

Theoretical Properties of RoPE

Efficient Computation (Eq. 34) — Exploiting Sparsity of $R^d_{\Theta,\mathbf{m}}$

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

∼ Relative Position by Construction

$q_m^T\, k_n \;=\; x^T\, W_q\, R^d_{\Theta,n-m}\, W_k\, x_n$ — the attention score depends only on the relative offset n−m, not absolute positions. No extra parameters needed.

⊗ Linear Attention Compatibility

Because RoPE is multiplicative and norm-preserving, it can be injected into linear (Performer) attention: multiply rotation matrix with outputs of the kernel functions $\varphi(q_m)\ and\ \varphi(k_n)$ (Eq. 19).

Linear Attention

$$\text{Attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})_m = \frac{\sum_{n=1}^{N} \phi(\boldsymbol{q}_m)^\intercal \varphi(\boldsymbol{k}_n)\boldsymbol{v}_n}{\sum_{n=1}^{N} \phi(\boldsymbol{q}_m)^\intercal \varphi(\boldsymbol{k}_n)},$$

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

Experiments: Pre-training Convergence

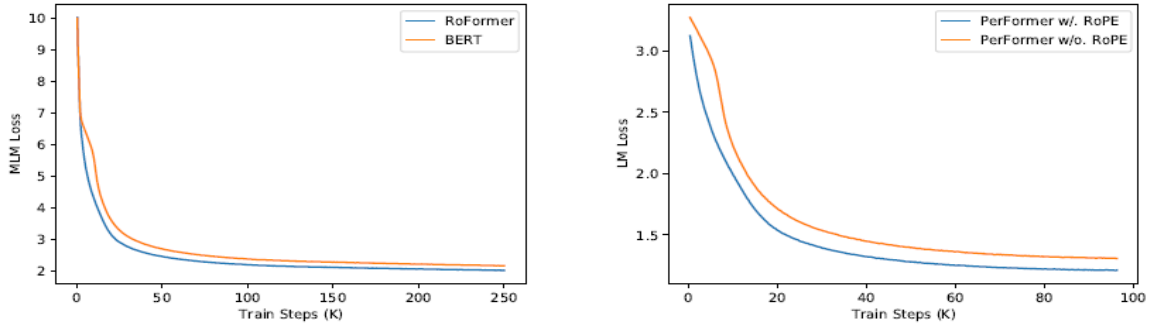


Figure 3: Evaluation of RoPE in language modeling pre-training. **Left**: training loss for BERT and RoFormer. **Right**: training loss for PerFormer with and without RoPE.

Bilingual Evaluation Understudy (BLEU):

It is a metric used to evaluate:

- machine translation
- text generation

By comparing the model's generated sentence with one or more human-written reference sentences.

Machine Translation — WMT 2014 En→De

Table 1: The proposed RoFormer gives better BLEU scores compared to its baseline alternative Vaswani et al. [2017] on the WMT 2014 English-to-German translation task Bojar et al. [2014].

Model	BLEU
Transformer-base Vaswani et al. [2017]	27.3
RoFormer	**27.5**

Setup

WMT 2014 (~4.5M sentence pairs) · BPE vocab 37k · Adam optimizer
Beam search (beam=4) · PyTorch + fairseq toolkit

Downstream Tasks: GLUE & Long-text Chinese Benchmark

Table 2: GLUE Benchmark Fine-tuning (vs BERT-base)

Model	MRPC	SST-2	QNLI	STS-B	QQP	MNLI
BERT	88.9	93.5	90.5	85.8	71.2	84.6/83.4
RoFormer	**89.5↑**	90.7	88.0	**87.0↑**	**86.4↑**	80.2/79.8

↑ RoFormer significantly outperforms BERT on MRPC, STS-B, and QQP

F1-score for MRPC and QQP dataset, Spearman correlation for STS-B, and accuracy for the remaining as the evaluation metric.

Table 5: CAIL2019-SCM (Chinese Long-text)

Model	Valid	Test
BERT-512	64.13%	67.77%
WoBERT-512	64.07%	68.10%
RoFormer-512	64.13%	68.29%
RoFormer-1024	**66.07%↑**	**69.79%↑**

↑ RoFormer-1024 surpasses WoBERT by +1.5%

Key Novelties of RoPE vs. Prior Position Encoding Methods

⟳ Multiplicative

Rotation applied to Q/K directly; prior works add p_i to embeddings before projection.

↔ Relative by Design

$q_m^T k_n$ naturally depends on (m−n) — no decomposition tricks needed.

∞ Sequence-Length Flexible

$\theta_i = 10000^{-2i/d}$ generalises to any length without retraining.

⊕ Linear-Attention Ready

Norm-preserving rotation embeds directly in Performer-style linear attention.

Conclusions & Limitations

1

Theoretical Foundation

RoPE is uniquely derived from the inner-product relative-position constraint — not an ad hoc design. The derivation shows rotation is the natural solution in 2D complex space.

2

Unified Absolute + Relative

Absolute positions are encoded via the rotation matrix; relative positions emerge automatically in the attention computation $q_m^T k_n = f(x_m, x_n, m-n)$.

3

Strong Empirical Performance

Faster convergence in pre-training; superior long-text performance (CAIL2019-SCM +1.5%); competitive GLUE results; improved BLEU on WMT 2014 En→De.

4

Wide Adoption

RoPE is now the de facto positional encoding in many LLMs (LLaMA, Mistral, GPT-NeoX, Falcon). Integrated into HuggingFace Transformers.

Limitations

- No formal explanation for why RoPE converges faster than other PE schemes.
- Requires hardware (V100 GPUs) for pre-training; computational cost not fully analyzed.
- Interaction between RoPE and different attention heads / architectures under-explored.

Su et al. (2023) · arXiv:2104.09864v5 · github.com/ZhuiyiTechnology/roformer · huggingface.co/docs/transformers/model_doc/roformer