Shuffle the words, same answer

Self-attention has **no built-in sense of order**. Feed it the same tokens in a different order and the outputs just come back in that same permuted order:

`the cat sat` $\equiv$ `sat cat the`

To a bag-of-tokens attention layer these are the *same* input. Every LLM has to **break this symmetry** and inject position somehow.

The classic answer: **add** a position vector to each token. RoPE's answer is unusually elegant: **rotate**, do not **add**. Position becomes an angle, and relative distance falls straight out of the query-key dot product.

RoPE in one line

rotate q by $m\theta$, k by $n\theta$

$\Rightarrow$ score depends only on the **relative** position $m-n$

touches only Q and K, nothing else

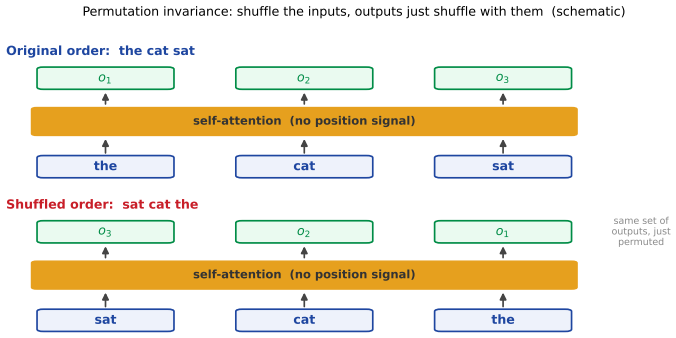
Outline

Attention is permutation-invariant

Position is not free – it has to be injected. The question is *how*.

Permutation invariance, concretely

Strip out position and self-attention treats a sentence as a **set**: permute the inputs and the outputs simply permute the same way.



The attention score is built from dot products $q_m^\top k_n$, which know *which* tokens are involved but not *where* they sit. Some position signal must be added, or the model is order-blind.

The classic fix: absolute position embeddings

Add a position vector p_i to each token embedding before projecting to q, k, v:

$$f_{q,k,v}(x_i, i) = W_{q,k,v}\,(x_i + p_i).$$

Two flavors, with **different** failure modes:

- **Learned** $p_i \in \{p_1, \ldots, p_L\}$: one trainable vector per index up to a **max length** L. Simple, but there is *no vector at all* for positions $> L$ – the model cannot run past its trained length.
- **Sinusoidal** $p_{i,2t} = \sin(i/10000^{2t/d})$, $p_{i,2t+1} = \cos(\cdot)$: defined for *every* position, so length is not hard-capped – but in practice it still **extrapolates poorly** past the lengths seen in training.

Both encode an **absolute** index. Neither makes the score a clean function of the **distance** $m - n$ – that relation only appears indirectly, if at all.

What we actually want

Intuitively, attention between two tokens should care about **how far apart** they are, not their absolute slots, and should **taper** at long range.

We want the query-key score to depend **only** on the relative position:

$$\langle f_q(x_m, m),\ f_k(x_n, n)\rangle = g(x_m, x_n,\ m-n) \qquad \text{(paper Eq. 11)}$$

- Depend on $m-n$, not on m and n separately.
- Decay as $|m-n|$ grows (far tokens matter less).
- Ideally **cheap**: no new terms in the attention block, defined at any position.

RoPE is engineered to hit exactly this target – and it does it by *rotation*.

Position as an angle

Intuition first (clock hands), then the two-line derivation, then numbers by hand.

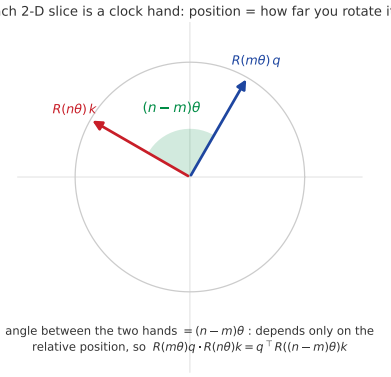
Intuition: each 2-D slice is a clock hand

Take a coordinate pair of q and treat it as a hand on a clock. **Position = how far you rotate the hand**: token at position m rotates by $m\theta$, token at position n by $n\theta$.

A dot product of two vectors depends on the **angle between them**. Here that angle is

$$n\theta - m\theta = (n - m)\theta,$$

the **difference** of the two rotations. So the score sees only the **relative** position, never the absolute one.

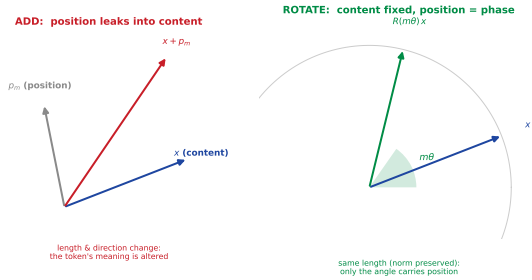


Rotating both hands by the same extra amount does not change the angle between them – which is exactly why absolute position cancels and only $m - n$ survives.

Why rotate instead of add?

Both inject position – but they do very different things to the vector:

- **Adding** p_i changes the token's *length and direction* – position **leaks into the content**. The score $(x_m+p_m)\cdot(x_n+p_n)$ has cross-terms that never reduce cleanly to $m-n$.
- **Rotating** is *length-preserving*: the content is untouched, position becomes a pure **phase**. Two rotated vectors' score depends *only* on the angle gap $(n-m)\theta$.



Rotation fixes *what* the token means and moves only *where* it sits – which is exactly why the dot product can isolate relative position.

The 2-D case: relative distance falls out

Rotate q by $m\theta$ and k by $n\theta$ with the 2×2 rotation matrix $R(\alpha) = \left(\begin{smallmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{smallmatrix}\right)$:

$$q_m = R(m\theta)\, q, \qquad k_n = R(n\theta)\, k.$$

Two facts about rotations: $R(\alpha)^\top = R(-\alpha)$ and $R(\alpha)R(\beta) = R(\alpha+\beta)$. So

$$q_m^\top k_n = \big(R(m\theta)q\big)^\top R(n\theta)k = q^\top R(-m\theta)R(n\theta)\, k = \boxed{q^\top R\big((n-m)\theta\big)\, k}.$$

The score depends **only** on $n - m$. (paper Eq. 12–16)

Absolute angles $m\theta$ and $n\theta$ went in; only their **difference** $(n-m)\theta$ comes out. Relative encoding, for free, inside the dot product.

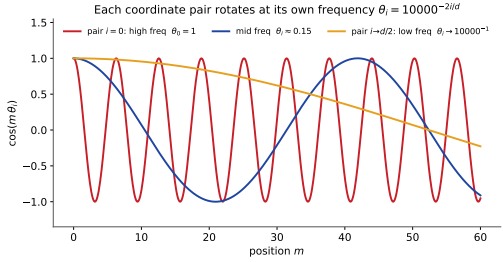
Generalize to d dimensions: one frequency per pair

Pair up the d coordinates into $d/2$ 2-D slices and rotate slice i at its **own** frequency:

$$\theta_i = 10000^{-2i/d}, \quad i = 0, \ldots, \frac{d}{2} - 1$$

- **High-frequency** pairs (small i) spin fast: they track *local* position.
- **Low-frequency** pairs (large i) spin slowly: they track *global* position.

Same spirit as the sinusoidal frequencies – but applied by **rotation**, not addition.



The convention $\theta_i = 10000^{-2i/d}$ is straight from the paper (§3.3). The base is fixed at **10000** throughout RoFormer.

By hand: same distance, same score

Take $q = (1, 0)$, $k = (0, 1)$, base angle $\theta = 30^\circ$. Positions $m = 2$, $n = 5$ (so $m\theta = 60^\circ$, $n\theta = 150^\circ$):

$$q_m = R(60^\circ)q = (0.50,\ 0.87), \qquad k_n = R(150^\circ)k = (-0.50,\ -0.87).$$

$$\text{score} = q_m^\top k_n = (0.50)(-0.50) + (0.87)(-0.87) = -0.25 - 0.75 = \mathbf{-1.0}.$$

Check with the boxed formula, $n - m = 3$ so the relative angle is $3\theta = 90^\circ$:

$$q^\top R(90^\circ)k = (1, 0)^\top(-1, 0) = \mathbf{-1.0}. \quad \checkmark$$

Now try $m = 0$, $n = 3$ (same $n - m = 3$): $q^\top R(90^\circ)k = -1.0$ *again*. Different absolute positions, **identical** score – because only $m - n$ enters.

Three properties that made RoPE win

Relative-in-absolute-clothing, proven long-term decay, and defined at any position.

Relative encoding in absolute clothing

RoPE applies the rotation **elementwise to Q and K only**, one position-dependent rotation per token. Yet the *score* it produces is purely relative.

- V **is untouched**, and so is the rest of the block – no extra attention terms, no learned relative-position table.
- Each token is rotated **once** by its own absolute angle; the relative structure emerges only when two rotated vectors meet in the dot product.
- Cheap and per-layer: a couple of elementwise sine/cosine multiplies (`apply_rotary_emb`), applied before attention.

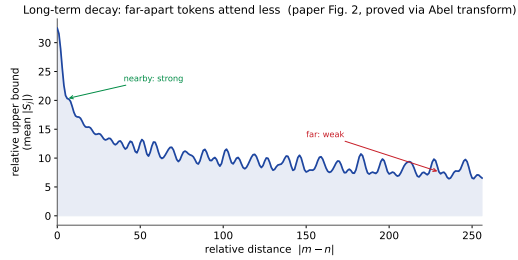
That is the trick in the name: **absolute** rotations applied to each token, giving a **relative** score – with none of the machinery earlier relative-position schemes needed.

Long-term decay is proved, not just hoped for

Summing over the $d/2$ frequencies, the paper bounds the score by a quantity that **shrinks as** $|m-n|$ **grows** (§3.4.3, via an Abel transformation):

$$\left| \sum_i h_i \, e^{i(m-n)\theta_i} \right| \leq \left(\max_i |h_{i+1} - h_i| \right) \sum_i |S_{i+1}|.$$

The multi-frequency sum makes far-apart tokens interfere and cancel, so their attention **decays**. This matches the intuition that distant tokens should matter less.



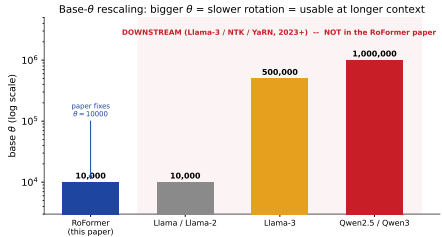
A single 2-D pair only gives $\cos((m-n)\theta)$, which oscillates. The **decay** comes from summing many frequencies at once (paper Fig. 2).

Sequence-length flexibility vs. context extension

What the paper claims. RoPE is defined by a formula at *any* position m, so unlike **learned** absolute embeddings it is not capped at a trained length L. The paper calls this **sequence-length flexibility**.

What the paper does NOT claim. It shows *no* zero-shot length extrapolation, and it fixes base $\theta = 10000$ throughout.

The trick that actually *extends* usable context – rescaling the base ($10000 \rightarrow 500000$) – is **later work**: Llama-3, NTK-aware scaling, YaRN (2023+).



Llama-3's $\theta = 500{,}000$ (deck 07) and Qwen3's $1e4 \rightarrow 1e6$ (deck 08) come from **after** this paper. "Flexibility" (defined everywhere) $\neq$ demonstrated **extrapolation**.

Modest paper, enormous adoption

The RoFormer benchmarks are small wins. The real story is what happened next.

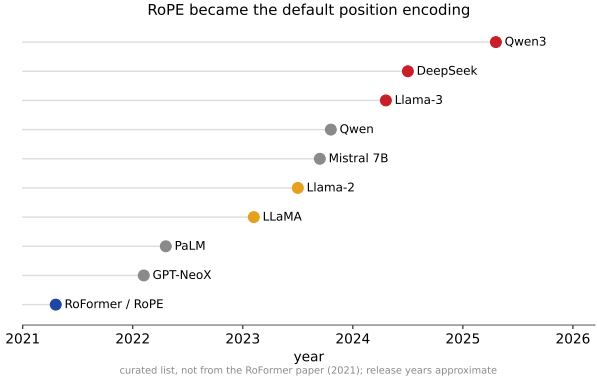
The numbers are modest; the adoption is not

RoFormer's own gains are **incremental**:

WMT14 En-De	BLEU
Transformer-base	27.3
RoFormer	**27.5**

On GLUE, RoFormer beats BERT on **3 of 6** tasks (big jump on QQP, smaller on MRPC and STS-B), and is behind on the rest. A solid result, not a landslide.

The lasting impact is that RoPE became the **default** position encoding across the field.



GPT-NeoX, PaLM, LLaMA / Llama-2 / Llama-3, Qwen, Mistral, DeepSeek all ship RoPE. (Adoption list is curated context, not from the 2021 paper.)

Recap

- **Why position at all:** self-attention is **permutation-invariant**; the score $q_m^\top k_n$ knows the tokens, not their places.
- **Why not just add:** absolute embeddings (learned or sinusoidal) encode an index, not a clean function of the distance $m - n$.
- **The idea:** rotate q by $m\theta$ and k by $n\theta$; then $q_m^\top k_n = q^\top R((n-m)\theta)k$ – depends **only** on $m - n$, touching only Q, K.
- **Multi-frequency** $\theta_i = 10000^{-2i/d}$: local and global pairs; long-term decay is **proved** (Abel transform, Fig. 2).
- **Flexibility:** defined at any position. Base-θ rescaling for long context ($10000 \to 500{,}000$) is **downstream** (Llama-3 / NTK / YaRN), not this paper.

Next: this is the position encoding inside *[LLM-7] Llama-3* ($\theta = 500\mathrm{k}$) and *[LLM-8] Qwen3* ($1\mathrm{e}4 \to 1\mathrm{e}6$); pushing context far also needs the efficient attention of *[LLM-15] FlashAttention*.