Deep Learning

Positional Encodings

Lucas Silveira Kupssinskü

Machine Learning Theory and Applications Laboratory
PUCRS

agosto de 2026

Visão Geral

Visão Geral

Por que a posição importa?

O mecanismo de *self-attention* calcula *scores* entre todos os pares:

Score de atenção

$$\text{score}(i, j) = \mathbf{q}_i^\top \mathbf{k}_j \;/\; \sqrt{d}$$

O produto escalar **não depende** dos índices i ou j, apenas dos **valores** de $\mathbf{q}$ e $\mathbf{k}$.

A *self-attention* (bidirecional) é invariante a permutação.

Sem informação posicional:
"`O gato sentou no tapete`" ≡ "`O tapete sentou no gato`"

O problema da permutação

Token	Sem PE	Com PE
"O"	mesma repr.	pos 0 → única
"gato"	mesma repr.	pos 1 → única
"sentou"	mesma repr.	pos 2 → única
"no"	mesma repr.	pos 3 → única
"tapete"	mesma repr.	pos 4 → única

Taxonomia de *Positional Encodings*

Absolute PE

- Vetor único por posição absoluta
- Sinusoidal (Vaswani et al., 2017)
- *Learned embeddings* (BERT, GPT)
- Somado ao *token embedding*

Relative PE

- Codifica distância $|i - j|$
- RoPE: rotação no espaço
- ALiBi: *bias* linear
- NoPE: máscara causal

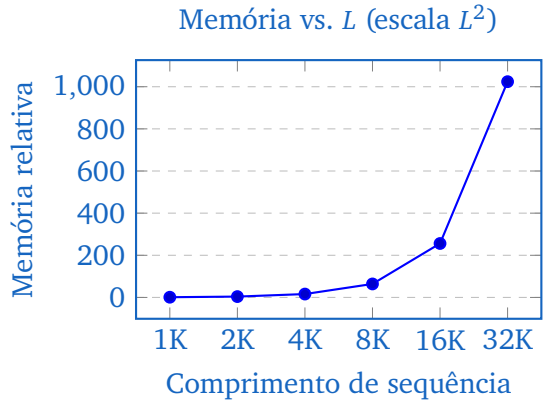
Extensão de Contexto

- *Fine-tune* para $L > L_{\text{train}}$
- *Position Interpolation*
- YaRN
- LongRoPE

Comprimento de Contexto e o Custo Quadrático

A atenção computa **todos os $L \times L$ pares** → memória e computação escalam como $\mathbf{O(L^2)}$.

- Dobrar $L \rightarrow 4\times$ mais memória para a matriz de atenção
- Um contexto de 100K *tokens* em um modelo 7B requer dezenas de GB só de atenção (em *fp16*, 2 bytes por valor)
- Maioria dos modelos treina com L = 2K–8K *tokens*
- Contextos longos desaceleram cada *forward pass* quadraticamente



Ilustrativo, memória de atenção escala como $O(L^2)$

Por que Extrapolação Importa: O Argumento Econômico

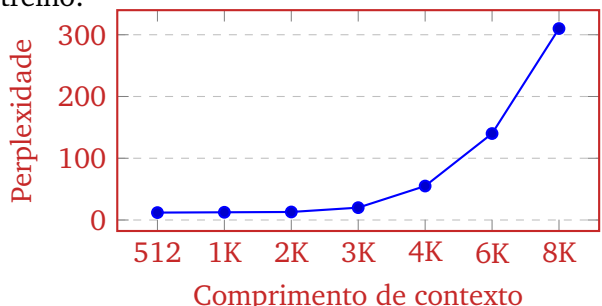
Extrapolação: usar o modelo em sequências mais longas que as vistas no treino ($L_{\text{test}} > L_{\text{train}}$).

- **O custo de treinamento é dominado pela atenção quadrática**
  - Treinar com $L = 2\text{K}$ custa $4\times$ menos que $L = 4\text{K}$
  - Treinar com $L = 8\text{K}$ custa $16\times$ menos que $L = 32\text{K}$
- **Se o PE puder extrapolar para sequências mais longas:**
  - **Treinamento mais leve com L pequeno, inferência com L maior sem perda de desempenho**
  - Sem necessidade de *fine-tuning* de contexto longo
- **Problema: a maioria dos PEs absolutos NÃO extrapola bem**
  - Esta aula: por que falham e como PEs relativos / extensão de contexto corrigem isso

O Desafio da Extrapolação

Um modelo treinado com $L_{\text{train}} = 2\text{K}$ encontra índices de posição **nunca vistos** quando $L_{\text{test}} > L_{\text{train}}$.

Perplexidade dispara além do horizonte de treino:



Ilustrativo, comportamento típico além do horizonte de treino

Como resolver:

- PE que codifique apenas distância relativa → Parte 2
- Reescalar/interpolar posições em tempo de teste → Parte 4
- Sem PE: deixar o modelo inferir → NoPE

Notação e Configuração

Símbolo	Significado
d	Dimensão do *embedding* (d_{model})
L	Comprimento da sequência
i	Posição do *token query* ($0 \leq i < L$)
j	Posição do *token key* ($0 \leq j < L$)
k	Índice de dimensão ($0 \leq k < d/2$)
$\mathbf{x}_i$	*Embedding* de entrada do *token* na posição i
$\mathbf{q}_i, \mathbf{k}_j$	Vetores *query* e *key* nas posições i, j
$\text{PE}(i)$	Vetor de *positional encoding* para posição i

PE Absoluto

$\mathbf{x}_i \leftarrow \mathbf{x}_i + \text{PE}(i)$ (somado ao *embedding* antes da atenção)

Visão Geral

PE Sinusoidal: Artigo

Referência

Vaswani, A. et al. “Attention Is All You Need.” NeurIPS 2017.[a]

[a] Ashish Vaswani et al. “Attention is all you need”. Em: NeurIPS. vol. 30. 2017.

- Introduziu a arquitetura *Transformer* (*encoder–decoder*)
- Propôs funções sinusoidais determinísticas como *positional encodings*
- Propriedade: nenhum parâmetro aprendido → custo extra zero no treino
- Em princípio, pode generalizar para comprimentos não vistos no treino
- Tornou-se o PE padrão para a maioria das variantes iniciais do *Transformer*

PE Sinusoidal: Por que Senoides?

Queremos uma "assinatura" de posição que (1) seja única para cada posição e (2) permita ao modelo recuperar *distâncias relativas* entre *tokens*.

Ideia: sobrepor ondas de várias frequências

Como na contagem binária, cada *bit* alterna a uma taxa diferente:

- Componentes **rápidos** (alta freq.) distinguem posições **vizinhas**
- Componentes **lentos** (baixa freq.) capturam a posição **global**

Senoides são o análogo **contínuo** dos *bits*.

Termos

Frequência ω: quão rápido a onda oscila.

Período (comprimento de onda) $= 2\pi/\omega$: a distância para a onda se repetir.

Por que seno e cosseno? Um deslocamento de δ vira uma **rotação** do par $(\sin, \cos)$, é isso que deixa a atenção ler *offsets* relativos (próximos slides).

PE Sinusoidal: Fórmula

Para posição i e índice de dimensão k (d = dimensão total do *embedding*):

Fórmula do *Positional Encoding* Sinusoidal

$$\mathrm{PE}(i, 2k) = \sin\left(\frac{i}{10000^{2k/d}}\right)$$

$$\mathrm{PE}(i, 2k{+}1) = \cos\left(\frac{i}{10000^{2k/d}}\right)$$

- Dimensão $2k$ usa **seno**; dimensão $2k{+}1$ usa **cosseno**, mesma frequência, defasagem de 90°
- Base 10000 (empírica): frequências altas (dim 0) a baixas (dim $d{-}1$)

Abreviação

$\omega_k = 10000^{-2k/d} \quad \Rightarrow \quad \mathrm{PE}(i, 2k) = \sin(i \cdot \omega_k), \quad \mathrm{PE}(i, 2k{+}1) = \cos(i \cdot \omega_k)$

PE Sinusoidal: Intuição

Analogia com contagem binária:

pos	bit 2 (lento)	bit 1 (médio)	bit 0 (rápido)
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1

→ PE sinusoidal é o análogo contínuo.

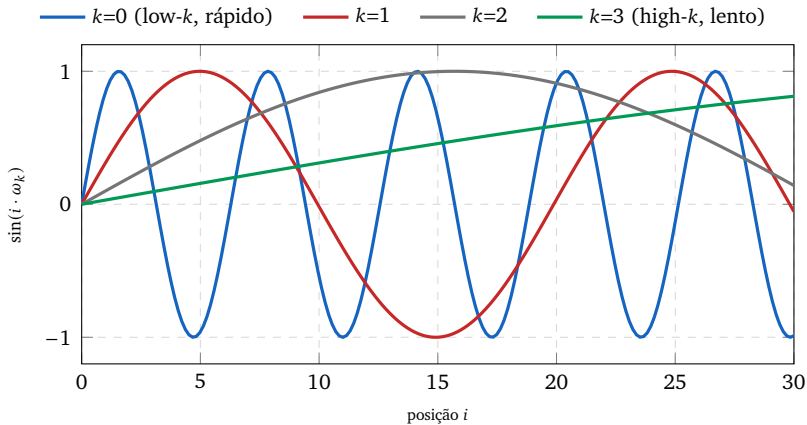
Cada dimensão = um "*canal de frequência*"

Low-k: oscilação rápida → posição fina
High-k: oscilação lenta → posição grosseira

Nomenclatura: low-k = índice k *pequeno* = **alta** frequência; **high-k** = índice k *grande* = **baixa** frequência.

$\mathrm{PE}(i+\delta)$ pode ser expresso como função linear de $\mathrm{PE}(i)$, permitindo ao modelo aprender *offsets* relativos a partir de PEs absolutos *(é a propriedade-chave; veja o slide de Propriedades)*.

PE Sinusoidal: Visualizando as Frequências ($d = 16$)



Cada k é um "relógio" em velocidade diferente: combinando frequências, cada posição tem uma assinatura única (análogo contínuo da contagem binária). **A seguir:** frase "*O gato sentou no tapete*" (L=5, d=4).

PE Sinusoidal: Matriz para “O gato sentou no tapete”

Matriz PE completa para a frase exemplo ($L = 5$, $d = 4$):

i (token)	dim 0	dim 1	dim 2	dim 3
0 (“O”)	0.000	1.000	0.000	1.000
1 (“gato”)	0.841	0.540	0.010	1.000
2 (“sentou”)	0.909	−0.416	0.020	1.000
3 (“no”)	0.141	−0.990	0.030	1.000
4 (“tapete”)	−0.757	−0.654	0.040	0.999

Dimensões 0–1 (ω_0=1) variam rapidamente com i; dimensões 2–3 (ω_1=0.01) quase não mudam nessa faixa.

Os próximos slides mostram passo a passo de onde vêm esses números.

Cálculo Manual, PE Sinusoidal, Passo 1: Configuração

Frase: "O gato sentou no tapete" $\rightarrow L = 5$, posições $i \in \{0, 1, 2, 3, 4\}$, $d = 4$

Calcular as frequências angulares $\omega_k = 10000^{-2k/d}$:

k	$2k/d$	Expoente	ω_k	Período $2\pi/\omega_k$
0	$0/4 = 0.0$	$10000^0 = 1$	$\omega_0 = 1.0000$	≈ 6.28
1	$2/4 = 0.5$	$10000^{0.5} = 100$	$\omega_1 = 0.0100$	≈ 628

Com $d = 4$ temos 2 pares de frequência ($k = 0, 1$).
k=0: oscilação rápida (período $\approx$ 6 tokens) | k=1: muito lenta (período $\approx$ 628 tokens).

Cálculo Manual, PE Sinusoidal, Passo 2: Preencher a Tabela

Aplicar $\mathrm{PE}(i, 2k) = \sin(i \cdot \omega_k)$ e $\mathrm{PE}(i, 2k{+}1) = \cos(i \cdot \omega_k)$ aos 5 tokens:

i (token)	dim 0 $\sin(i \cdot \omega_0)$	dim 1 $\cos(i \cdot \omega_0)$	dim 2 $\sin(i \cdot \omega_1)$	dim 3 $\cos(i \cdot \omega_1)$
0 (“O”)	$\sin(0) = 0.000$	$\cos(0) = 1.000$	$\sin(0) = 0.000$	$\cos(0) = 1.000$
1 (“gato”)	$\sin(1) = 0.841$	$\cos(1) = 0.540$	$\sin(0.01) = 0.010$	$\cos(0.01) = 1.000$
2 (“sentou”)	$\sin(2) = 0.909$	$\cos(2) = -0.416$	$\sin(0.02) = 0.020$	$\cos(0.02) = 1.000$
3 (“no”)	$\sin(3) = 0.141$	$\cos(3) = -0.990$	$\sin(0.03) = 0.030$	$\cos(0.03) = 1.000$
4 (“tapete”)	$\sin(4) = -0.757$	$\cos(4) = -0.654$	$\sin(0.04) = 0.040$	$\cos(0.04) = 0.999$

Os valores reproduzem exatamente a matriz prevista no slide anterior. As dimensões 2 e 3 quase não mudam: o modelo usa essas dimensões para codificar posição **grosseira** (milhares de passos).

Cálculo Manual, Passo 3: Somar PE ao *Embedding*

***Token embeddings* de "O gato sentou no tapete":**

i (token)	$x[0]$	$x[1]$	$x[2]$	$x[3]$
0 ("O")	0.500	−0.200	0.100	0.800
1 ("gato")	0.300	0.600	−0.400	0.200
2 ("sentou")	−0.100	0.400	0.700	−0.300
3 ("no")	0.600	−0.500	0.200	0.100
4 ("tapete")	−0.200	0.300	−0.600	0.500

Após somar o PE sinusoidal:

i	$x[0]$ + PE[0]	$x[1]$ + PE[1]	$x[2]$ + PE[2]	$x[3]$ + PE[3]
0	0.500 + 0.000 = 0.500	−0.200 + 1.000 = 0.800	0.100 + 0.000 = 0.100	0.800 + 1.000 = 1.800
1	0.300 + 0.841 = 1.141	0.600 + 0.540 = 1.140	−0.400 + 0.010 = −0.390	0.200 + 1.000 = 1.200
2	−0.100 + 0.909 = 0.809	0.400 − 0.416 = −0.016	0.700 + 0.020 = 0.720	−0.300 + 1.000 = 0.700
3	0.600 + 0.141 = 0.741	−0.500 − 0.990 = −1.490	0.200 + 0.030 = 0.230	0.100 + 1.000 = 1.100
4	−0.200 − 0.757 = −0.957	0.300 − 0.654 = −0.354	−0.600 + 0.040 = −0.560	0.500 + 0.999 = 1.499

✓ A mesma rede processa os 5 *tokens*; a posição fica codificada nos valores.

PE Sinusoidal: Propriedades

- **Determinístico**: nenhum parâmetro aprendido, idêntico entre execuções
- **Limitado**: todos os valores em $[-1, 1]$, estável com normalização
- **Único por posição**: cada i gera um vetor PE distinto
- ***Offsets* relativos são lineares**: $\text{PE}(i + \delta) = M(\delta) \cdot \text{PE}(i)$ para uma matriz fixa M
  - Permite ao modelo aprender distância relativa a partir de PE absoluto
- **Teoricamente generalizável**: posições além do treino existem na mesma curva
  - Na prática, a generalização é limitada → ver próximo slide

Learned Absolute PE

- **BERT, GPT iniciais:** PE é uma matriz de *embeddings* treinável $E \in \mathbb{R}^{L_{\max} \times d}$
- Consulta por posição: $\mathrm{PE}(i) = E[i, :]$
- **Vantagens:** adapta-se à distribuição dos dados; sem frequências manuais
- **Desvantagens:**
  - Limite rígido em $L_{\max}$: posições além simplesmente não existem
  - *Overfits* ao comprimento de treino; pior OOD (*out-of-distribution*, fora da distribuição de treino) que sinusoidal
  - Adiciona $L_{\max} \times d$ parâmetros ao modelo
- **Modelos modernos migraram para PE relativo (RoPE, ALiBi) →**

Limitação do PE Absoluto

- O modelo vê posições absolutas durante o treino: $i = 0, 1, \ldots, L_{\text{train}} - 1$
- **Posições $i \geq L_{\text{train}}$ são *out-of-distribution***: nunca vistas no treino
- As frequências sinusoidais não extrapolam os padrões de atenção de forma confiável
- *Learned absolute* PE (BERT) é ainda pior, sem forma fechada para novas posições
- **Problema central: posição codificada independentemente do contexto**
  - "gato" na posição 0 e na posição 100 recebem vetores PE diferentes, mesmo que a relação semântica seja a mesma
- **Isso motiva os *positional encodings* relativos →**

De Absoluto para Relativo

O problema fundamental do PE absoluto:

Dois *tokens* “gato” à distância 2 de “sentou” deveriam ter o mesmo *score* de atenção, independentemente de estarem nas posições $(0, 2)$ ou $(1000, 1002)$.

PE relativo codifica $j - i$ diretamente:

- O *score* de atenção depende apenas do *offset* relativo $j - i$, não de i ou j individualmente
- Modelos treinados com comprimento L podem generalizar para qualquer comprimento
- Três estratégias: **RoPE** (rotação), **ALiBi** (*bias* aditivo), **NoPE** (implícito)

Visão Geral

Ideia Central: *Positional Encodings* Relativos

Objetivo: fazer o *score* de atenção depender apenas de $(j - i)$.

Formulação

$\text{score}(i, j) = f(\mathbf{q}_i, \mathbf{k}_j, j{-}i)$ onde f depende de $(j{-}i)$, não de i ou j separadamente

Três estratégias:

- **RoPE**: rotacionar $\mathbf{q}$ e $\mathbf{k}$ por ângulo dependente da posição; produto escalar codifica $j{-}i$
- **ALiBi**: somar penalidade linear $-m \cdot |i{-}j|$ aos *logits* de atenção
- **NoPE**: sem PE; posição implícita via máscara causal

Todas alcançam generalização de comprimento em graus variados.

RoPE: Artigo

Referência

Su, J. et al. “RoFormer: Enhanced Transformer with Rotary Position Embedding.” Neurocomputing 2024.[a]

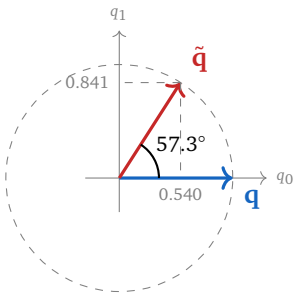
[a] Jianlin Su et al. “RoFormer: Enhanced transformer with rotary position embedding”. Em: Neurocomputing 568 (2024), p. 127063.

- Codificar posição **rotacionando** os vetores *query* e *key*
- Rotação é operação linear, compatível com qualquer arquitetura *Transformer*
- O produto interno $\mathbf{q}_i^\top \mathbf{k}_j$ depende apenas de $(j - i)$ após a rotação ✓
- Adotado por LLaMA, Mistral, Falcon, Gemma e maioria dos LLMs *open-source*
- Base para métodos de extensão de contexto (YaRN, LongRoPE, etc.)

RoPE: Visualizando a Rotação 2D

Intuição do RoPE: posição = ângulo de rotação (a álgebra vem a seguir).

Bloco k=0: rotacionar $\mathbf{q} = (1, 0)$ por $i \cdot \theta_0 = 1.0$ rad



Bloco k=0: *query* em $i = 1$

Multiplicação matricial

$$\begin{pmatrix} \cos 1.0 & -\sin 1.0 \\ \sin 1.0 & \cos 1.0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.540 \\ 0.841 \end{pmatrix}$$

- O vetor é rotacionado 57.3° no sentido anti-horário
- **Rotação preserva a magnitude:** $|\mathbf{q}| = |\tilde{\mathbf{q}}| = 1$ ✓
- $\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{k}}_j$ passará a depender só de $(j-i) \cdot \theta_0$ *(provado a seguir)*

RoPE: Ideia Central

Em vez de somar PE ao *embedding*, **rotacionar Q e K**:

Rotação e *Score*

$$\tilde{\mathbf{q}}_i = R_\theta(i) \cdot \mathbf{q}_i \qquad \tilde{\mathbf{k}}_j = R_\theta(j) \cdot \mathbf{k}_j$$

$$\text{score}(i,j) = \tilde{\mathbf{q}}_i^\top \tilde{\mathbf{k}}_j = \mathbf{q}_i^\top R_\theta(i)^\top R_\theta(j)\, \mathbf{k}_j = \mathbf{q}_i^\top R_\theta(j-i)\, \mathbf{k}_j$$

Como matrizes de rotação satisfazem $R_\theta(i)^\top = R_\theta(-i)$ (rotação no sentido oposto), temos $R_\theta(i)^\top R_\theta(j) = R_\theta(-i)\, R_\theta(j) = R_\theta(j-i)$, **o *score* depende apenas do *offset* relativo $j - i$.**

- $R_\theta(m)$ é uma matriz de rotação bloco-diagonal parametrizada pela posição m
- Cada bloco 2D rotaciona um par de dimensões por ângulo $m \cdot \theta_k$
- **PE NÃO é somado à entrada, é aplicado dentro do *attention head***

RoPE: O Bloco de Rotação 2D

Para um par de dimensões (q_0, q_1) na posição i:

Matriz de rotação 2D

$$R_\theta(i) = \begin{pmatrix} \cos(i\theta) & -\sin(i\theta) \\ \sin(i\theta) & \cos(i\theta) \end{pmatrix}$$

Par rotacionado: $\left[\cos(i\theta)\, q_0 - \sin(i\theta)\, q_1,\ \ \sin(i\theta)\, q_0 + \cos(i\theta)\, q_1\right]$

A mesma rotação é aplicada ao par correspondente de *key*, no ângulo $j \cdot \theta$.

O produto escalar equivale a uma rotação por $(j-i) \cdot \theta$, depende só do *offset*.

Expansão

Somando os dois produtos componente a componente, os termos cruzados de sin/cos recombinam-se em $\cos((j-i)\theta)$ e $\sin((j-i)\theta) \rightarrow$ depende apenas de $(j-i)$. *Derivação completa no próximo slide.*

RoPE: Fórmula Completa (d Dimensões)

Para d dimensões, aplicar $d/2$ blocos de rotação independentes:

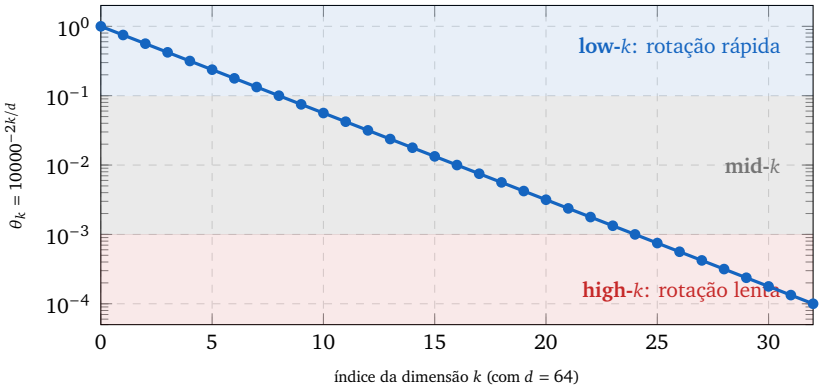
Frequências e rotação bloco-diagonal

$$\theta_k = 10000^{-2k/d} \qquad k = 0, 1, \ldots, d/2-1$$

$$R_\theta(i) = \text{block-diag}\big(R(i\theta_0),\ R(i\theta_1),\ \ldots,\ R(i\theta_{d/2-1})\big)$$

- Mesma base 10000 do PE sinusoidal: frequências cobrem muitas ordens de magnitude
- *Low-k* ($\theta_k \approx 1$): rotação rápida → posição relativa fina
- *High-k* ($\theta_k \approx 0$): rotação lenta → posição relativa grosseira
- Eficiente: tabelas cos/sin pré-computadas; fundido com projeção Q·K

RoPE: Espectro de Frequências



Cobertura de ~4 ordens de magnitude em frequência. **Diversidade espectral é a base do sucesso do RoPE**: cada bloco 2D rotaciona numa velocidade diferente, permitindo codificar posições relativas finas e grosseiras simultaneamente.

RoPE, Por que Funciona: Esboço de Prova

Afirmação: $\tilde{\mathbf{q}}_i^\top \tilde{\mathbf{k}}_j$ depende apenas de $(j - i)$.

Foco em um bloco 2D com frequência θ:

Derivação

$$\tilde{\mathbf{q}}_i = \big(\cos(i\theta)\, q_0 - \sin(i\theta)\, q_1,\ \ \sin(i\theta)\, q_0 + \cos(i\theta)\, q_1\big)$$

$$\tilde{\mathbf{k}}_j = \big(\cos(j\theta)\, k_0 - \sin(j\theta)\, k_1,\ \ \sin(j\theta)\, k_0 + \cos(j\theta)\, k_1\big)$$

$$\tilde{\mathbf{q}}_i \cdot \tilde{\mathbf{k}}_j = (q_0 k_0 + q_1 k_1)\cos\big((j-i)\theta\big) + (q_1 k_0 - q_0 k_1)\sin\big((j-i)\theta\big)$$

$$\rightarrow \text{depende apenas de } j - i \ \checkmark$$

- Cada bloco contribui um par $(\cos, \sin)$ que depende de $(j - i)$
- **Somando $d/2$ blocos: *score* total é função de $(\mathbf{q}, \mathbf{k}, j{-}i)$**

Cálculo Manual, RoPE, Passo 1: Configuração

Parâmetros: $d = 4$, $\theta_0 = 1.0$, $\theta_1 = 0.01$
Query em $i = 1$: $\mathbf{q} = [1, 0, 1, 0]$ *Key* em $j = 3$: $\mathbf{k} = [0, 1, 0, 1]$

Ângulos de rotação:

Bloco	θ_k	Ângulo i=1 (*query*)	Ângulo j=3 (*key*)
k=0	1.0000	$1 \times 1.00 = 1.0000$ rad	$3 \times 1.00 = 3.0000$ rad
k=1	0.0100	$1 \times 0.01 = 0.0100$ rad	$3 \times 0.01 = 0.0300$ rad

Valores trigonométricos:

ângulo	cos	sin
1.0000	0.540	0.841
3.0000	−0.990	0.141
0.0100	1.000	0.010
0.0300	1.000	0.030

Cálculo Manual, RoPE, Passo 2: Aplicar Rotações

Rotacionar $\mathbf{q} = [1, 0, 1, 0]$ em $i = 1$:

Bloco	Entrada	Ângulo	Fórmula	Resultado
k=0	$(1, 0)$	1.00 rad	$\cos\cdot 1 - \sin\cdot 0,\ \sin\cdot 1 + \cos\cdot 0$	$(0.540, 0.841)$
k=1	$(1, 0)$	0.01 rad	$\cos\cdot 1 - \sin\cdot 0,\ \sin\cdot 1 + \cos\cdot 0$	$(1.000, 0.010)$

$\rightarrow \tilde{\mathbf{q}}(i{=}1) = [0.540,\ 0.841,\ 1.000,\ 0.010]$

Rotacionar $\mathbf{k} = [0, 1, 0, 1]$ em $j = 3$:

Bloco	Entrada	Ângulo	Fórmula	Resultado
k=0	$(0, 1)$	3.00 rad	$\cos\cdot 0 - \sin\cdot 1,\ \sin\cdot 0 + \cos\cdot 1$	$(-0.141, -0.990)$
k=1	$(0, 1)$	0.03 rad	$\cos\cdot 0 - \sin\cdot 1,\ \sin\cdot 0 + \cos\cdot 1$	$(-0.030, 1.000)$

$\rightarrow \tilde{\mathbf{k}}(j{=}3) = [-0.141,\ -0.990,\ -0.030,\ 1.000]$

Cálculo Manual, RoPE, Passo 3: Produto Escalar e Verificação

score(i=1, j=3) = $\tilde{\mathbf{q}}(1) \cdot \tilde{\mathbf{k}}(3)$:

Cálculo

$= 0.540 \cdot (-0.141) + 0.841 \cdot (-0.990) + 1.000 \cdot (-0.030) + 0.010 \cdot 1.000 = \mathbf{-0.929}$

Verificação: depende apenas de $(j - i) = 2$?

Rotacionar o mesmo $\mathbf{q}$ em $i = 0$ e o mesmo $\mathbf{k}$ em $j = 2$ (*offset* relativo $= 2$):

$\tilde{\mathbf{q}}(0) = [1, 0, 1, 0]$ (rotação por 0: identidade)

$\tilde{\mathbf{k}}(2)$: bloco $0 \rightarrow (-0.909, -0.416)$; bloco $1 \rightarrow (-0.020, 1.000)$

$\tilde{\mathbf{k}}(2) = [-0.909, -0.416, -0.020, 1.000]$

Verificação

score $= 1 \cdot (-0.909) + 0 \cdot (-0.416) + 1 \cdot (-0.020) + 0 \cdot 1.000 = \mathbf{-0.929}$ ✓ Mesmo *score*!

ALiBi: Artigo

Referência

Press, O. et al. “Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation.” ICLR 2022.[a]

[a] Ofir Press, Noah A Smith e Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. Em: ICLR. 2022.

- Nenhuma modificação nos *embeddings*: zero vetores posicionais adicionados
- Em vez disso: subtrair uma penalidade linear dos *logits* de atenção
- Penalidade proporcional a $|i - j|$: *tokens* distantes recebem penalidade maior
- Cada *head* h tem uma inclinação diferente $m_h \longrightarrow$ espectro de atenção local a global
- **Extrapolação *zero-shot*: treinado em $L = 1$K, funciona em $L = 8$K**

ALiBi: Ideia Central

RoPE extrapola apenas com ajustes (interpolação/*fine-tuning*); ALiBi busca extrapolação *zero-shot* por outro caminho, um viés direto na atenção.

Atenção padrão (*pre-softmax*):

$$A(i,j) = \mathbf{q}_i^\top \mathbf{k}_j \;/\; \sqrt{d}$$

Atenção ALiBi (*pre-softmax*):

$$A_h(i,j) = \mathbf{q}_i^\top \mathbf{k}_j \;/\; \sqrt{d} \;-\; m_h \cdot |i-j|$$

A penalidade $m_h \cdot |i-j|$ cresce com a distância, *tokens* distantes são ponderados para baixo.

Diferentes *heads* h têm inclinações m_h diferentes → espectro de atenção local a global.

- **Sem vetores de posição, custo de PE é zero**
- Em sequências longas de teste, o modelo simplesmente aplica penalidades maiores, permanece *in-distribution*

ALiBi: Inclinações por *Head*

As inclinações formam uma sequência geométrica sobre os n *attention heads*:

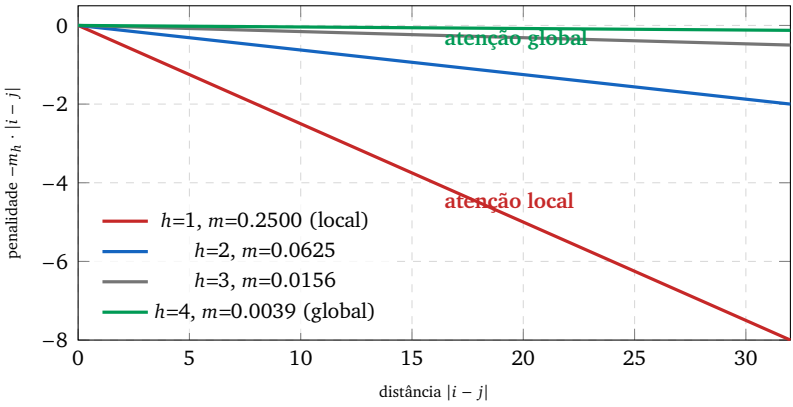
$$m_h = 2^{-8h/n} \qquad h = 1, 2, \ldots, n$$

O expoente 8 é uma escolha empírica de [7] (calibra a faixa de inclinações), sem derivação teórica.

n *heads*	h=1 (mais íngreme)	h=2	h=3	h=4 (mais plano)
4	$2^{-2} = 0.2500$	$2^{-4} = 0.0625$	$2^{-6} = 0.0156$	$2^{-8} = 0.0039$
8	$2^{-1} = 0.5000$	$2^{-2} = 0.2500$	$2^{-3} = 0.1250$	$2^{-4} = 0.0625$

- Inclinações íngremes (m grande) → *head* atende principalmente localmente
- Inclinações planas (m pequeno) → *head* atende a longas distâncias
- Juntos, os *heads* cobrem amplo espectro de raios de atenção
- Inclinações são hiperparâmetros fixos, não aprendidos

ALiBi: Curvas de Decaimento por *Head*



Inclinações íngremes → *head* atende principalmente localmente. Inclinações planas → *head* mantém atenção em distâncias longas.

Juntos, os n *heads* cobrem todo o espectro de raios de atenção, do local ao global.

Cálculo Manual, ALiBi, Passo 1: Configuração e Inclinações

Parâmetros: $n_{\text{heads}} = 4$, seq_len $= 4$ (posições $i, j \in \{0, 1, 2, 3\}$)

Calcular inclinações (foco nos *heads* 1 e 2):

Head h	Fórmula $2^{-8h/n}$	Valor
1	$2^{-8\cdot 1/4} = 2^{-2}$	$m_1 = 0.2500$
2	$2^{-8\cdot 2/4} = 2^{-4}$	$m_2 = 0.0625$
3	$2^{-8\cdot 3/4} = 2^{-6}$	$m_3 = 0.0156$
4	$2^{-8\cdot 4/4} = 2^{-8}$	$m_4 = 0.0039$

Cálculo Manual, ALiBi, Passo 2: Matrizes de *Bias*

Matriz $|i-j|$ crua (causal: triângulo superior mascarado):

$i\backslash j$	j=0	j=1	j=2	j=3
i=0	0	$-\infty$	$-\infty$	$-\infty$
i=1	1	0	$-\infty$	$-\infty$
i=2	2	1	0	$-\infty$
i=3	3	2	1	0

$\times\, m_1 = 0.25 \rightarrow$ *Bias* do *head* 1:

$i\backslash j$	j=0	j=1	j=2	j=3
i=0	0	$-\infty$	$-\infty$	$-\infty$
i=1	-0.25	0	$-\infty$	$-\infty$
i=2	-0.50	-0.25	0	$-\infty$
i=3	-0.75	-0.50	-0.25	0

Posições próximas ($|i-j|$ pequeno) recebem penalidade menor que posições distantes. *Head* 2 ($m = 0.0625$) teria penalidades 4× menores → raio de atenção efetivo mais longo.

Cálculo Manual, ALiBi, Passo 3: Aplicar *Bias* e *Softmax*

Logits brutos para linha i=2: [2.0, 1.5, 3.0, −]

Head	*Scores* brutos	*Bias* ALiBi	Após *bias*	softmax ≈
h_1 (m=0.25)	[2.0, 1.5, 3.0]	[−0.50, −0.25, 0]	[1.50, 1.25, 3.0]	[0.16, 0.13, 0.71]
h_2 (m=0.0625)	[2.0, 1.5, 3.0]	[−0.125, −0.063, 0]	[1.88, 1.44, 3.0]	[0.21, 0.14, 0.65]

Head 1 (inclinação maior): mais peso na posição j=2 (o *token* imediatamente anterior).
Head 2 (inclinação menor): distribui atenção mais amplamente entre $j \in \{0, 1, 2\}$.

Em tempo de teste com $L > L_{\text{train}}$, o *bias* simplesmente fica maior para distâncias maiores → nenhum problema de OOD.

ALiBi: Propriedades

- Zero custo no *embedding*: nenhum vetor adicionado à entrada
- **Extrapolação *zero-shot*: treinado em L=1K, funciona em L=8K sem *fine-tuning***
- Modificação simples: apenas 1–2 linhas de código na atenção
- Diferentes *heads* atendem a diferentes faixas automaticamente
- **Ressalva: decaimento linear pode não ser adequado para todas as tarefas**
- Inclinações são fixas, não aprendidas por tarefa
- Não é trivialmente compatível com extensão de contexto baseada em RoPE

NoPE: Artigo

Referência

Kazemnejad, A. et al. “The Impact of Positional Encoding on Length Generalization in Transformers.” NeurIPS 2023.[a]

[a] Amirhossein Kazemnejad et al. “The impact of positional encoding on length generalization in transformers”. Em: NeurIPS. vol. 36. 2023.

- Estudo sistemático: comparar sinusoidal, RoPE, ALiBi e **nenhum PE**
- **Achado surpreendente: modelos sem PE generalizam para sequências mais longas**
- A máscara causal de atenção codifica implicitamente a ordem dos *tokens*
- NoPE superou todos os PEs explícitos testados em *benchmarks* de generalização
- **Ressalvas: resultados dependem fortemente do tipo de tarefa e tamanho do modelo**

NoPE: Ideia Central

Não usar nenhum *positional encoding*. Deixar a máscara causal fazer o trabalho.

- Em um *decoder* autorregressivo, a máscara de atenção é estritamente triangular inferior
- *Token* na posição i só pode atender a posições $j \leq i$
- **O padrão de quais *tokens* podem ser atendidos é, em si, informação posicional**
- Cada camada vê um "horizonte de histórico" diferente
- Como a máscara muda suavemente com o comprimento, sequências mais longas não são OOD

Por que isso não contradiz o início da aula

A invariância a permutação vale para a atenção **bidirecional**. A **máscara causal** já quebra essa simetria (cada posição i vê um conjunto diferente de *tokens*), então o *decoder-only* carrega um sinal de ordem mesmo sem PE. Não vale para *encoder-only* (BERT).

NoPE: Por que Remover PE Ajuda na Generalização de Comprimento

- **PE absoluto: posições $> L_{\text{train}}$ são *out-of-distribution***
- PE relativo (RoPE): frequências calibradas para o comprimento de treino, pode degradar
- **NoPE: nenhum sinal posicional adicionado → nada se torna OOD em comprimentos maiores**
- O único *inductive bias* posicional é o padrão da máscara, que escala naturalmente
- **Ressalva: o modelo precisa inferir posição puramente a partir do contexto**
  - Funciona bem para tarefas de linguagem com pistas contextuais ricas
  - Falha em tarefas que exigem posição absoluta (indexação, aritmética)

NoPE: Limitações

- **Depende de máscara causal, não se aplica a modelos *encoder-only* (BERT)**
- Não lida com tarefas que exigem posição absoluta
- Convergência mais lenta, o modelo precisa aprender padrões posicionais apenas dos dados
- Empiricamente pior que RoPE em alguns *benchmarks* de contexto longo (SCROLLS, LongBench)
- Métodos de extensão de contexto para RoPE não se aplicam diretamente
- **NoPE é um *baseline* útil e forte generalizador, ainda não é o padrão para grandes LLMs**

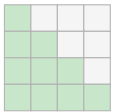
NoPE, Discussão: Máscara Causal como PE Implícito

Máscara causal para seq_len = 4:

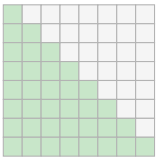
$i \backslash j$	j=0	j=1	j=2	j=3
i=0	1	—	—	—
i=1	1	1	—	—
i=2	1	1	1	—
i=3	1	1	1	1

- Cada linha tem padrão de máscara único
- **Softmax normaliza sobre i+1 tokens visíveis:** a magnitude **média** do peso de atenção escala como $\bar{\alpha}_{ij} \approx \frac{1}{i+1}$ → dá ao modelo um sinal indireto de i
- i=0 vê apenas ele mesmo; i=3 vê todos os 4
- Esse padrão muda previsivelmente com $L \rightarrow$ generaliza para $L > L_{\text{train}}$
- Cada camada empilha esse sinal, modelos profundos ganham representação posicional implícita rica

NoPE: Padrão Causal Escalando com L



$L = 4$



$L = 8$

$L = 16$

- O padrão triangular é **auto-similar**: escala continuamente com L
- **Nenhuma quebra de distribuição em $L > L_{\text{train}}$ → por isso NoPE generaliza**
- Cada linha i permanece única (vê exatamente i+1 posições passadas), independente de L

Comparação: Todos os Métodos de PE

Método	Aprendido?	Modifica…	Extrapolação	Custo
Sinusoidal	Não	*Input embedding*	Pobre (OOD)	Nenhum
Learned Abs.	Sim	*Input embedding*	Falha (sem novos pos.)	$L \times d$ parâmetros
RoPE	Não	Rotações Q, K	Moderada	Tabela cos/sin
ALiBi	Não	*Logits* de atenção	Boa (*zero-shot*)	Desprezível
NoPE	Não	Nada	Boa (apenas máscara)	Nenhum

RoPE é o padrão *de facto* para grandes LLMs (equilíbrio entre qualidade e extensibilidade).
ALiBi oferece a melhor extrapolação *zero-shot*.
NoPE é um *baseline* forte para generalização de comprimento.

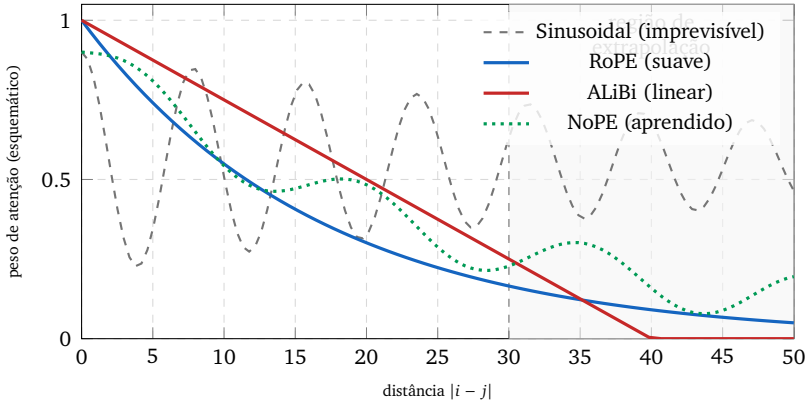
Padrões de Peso de Atenção por Método

Como cada método modela a atenção em função da distância $|i - j|$:

Método	**Fórmula do *score***	**Efeito em $\lvert i-j \rvert$ grande**
Sinusoidal	$\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d}$ (*rotated embed*)	Imprevisível (absoluto)
RoPE	$f(\mathbf{q}_i, \mathbf{k}_j, j{-}i)$ via rotação	Decaimento suave (dependente de freq.)
ALiBi	$\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d} - m \cdot \lvert i-j \rvert$	Decaimento linear garantido por *head*
NoPE	$\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d}$ (sem PE)	Aprendido pelo modelo via contexto

- ALiBi oferece o decaimento mais previsível, mais fácil de raciocinar sobre extrapolação
- RoPE tem decaimento dependente de frequência, adaptável via reescalonamento (LongRoPE)
- NoPE depende inteiramente do modelo aprender consciência posicional dos dados

Padrões de Decaimento: Comparação Visual

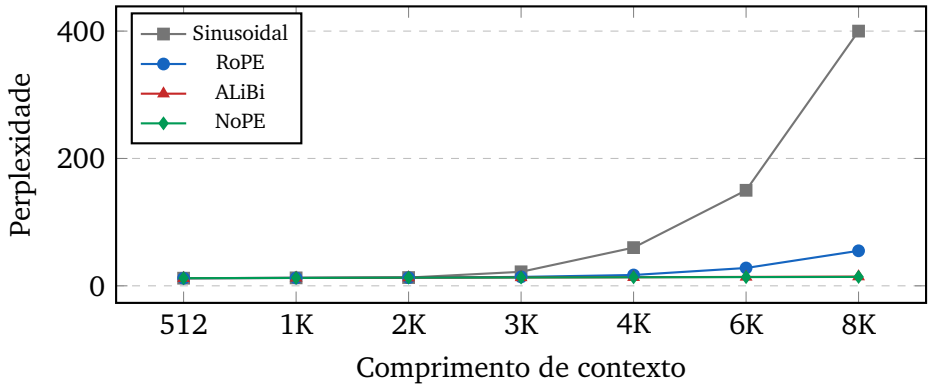


ALiBi: decaimento mais previsível → extrapolação garantida. **RoPE**: decaimento suave mas dependente de frequência. Sinusoidal tende a oscilar; NoPE depende do que o modelo aprendeu.

Curvas esquemáticas para comparação qualitativa, não valores experimentais.

Resultados Empíricos: Perplexidade vs. Comprimento de Contexto

Desempenho relativo em modelagem de linguagem (menor = melhor). Todos treinados em $L = 2048$:



Ilustrativo; tendências relativas de Press et al. (2022), Kazemnejad et al. (2023)

Transição: Do Design de PE para Desafios de Uso

Agora temos modelos com bons *positional encodings*.
Mas duas questões-chave permanecem:

1. Contexto longo realmente ajuda?

Lost in the Middle
→ Parte 3

2. Podemos estender contexto além do treino?

LongRoPE, YaRN, PI
→ Parte 4

Visão Geral

Perplexidade: Definição

Definição: mede o quão "surpreso" o modelo fica ao ver o texto.

$$\mathrm{PPL}(w_1, \ldots, w_N) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i \mid w_1, \ldots, w_{i-1})\right)$$

Vantagens

- Simples de calcular em qualquer corpus
- Permite comparar PEs em condições iguais
- Correlaciona com qualidade do modelo de linguagem

Atenção

PPL é uma **média sobre todos os tokens**.

Tokens iniciais são previstos com contexto curto; tokens finais com contexto completo.

Contribuições locais e distantes ficam *misturadas*.

Perplexidade Não Captura Uso Real do Contexto Longo

O problema central

Um modelo com **PPL baixa** pode **ignorar completamente** o contexto distante, e ainda assim parecer "bom".

Por quê?

- A maioria dos tokens é prevista corretamente a partir de contexto **local** (bigramas, padrões sintáticos)
- PPL *dilui* falhas de longo alcance na média global
- Um modelo com PPL = 14 a 8K tokens pode **nunca usar** o que foi dito nos primeiros 6K

O que PPL mede

Qualidade de *previsão do próximo token* em média sobre todos os tokens

O que PPL *não* mede

Se o modelo *utiliza* informações de posições distantes ao gerar uma resposta

$\Rightarrow$ Precisamos de avaliações baseadas em tarefas de recuperação de longo alcance.

Lost in the Middle: Artigo

Referência

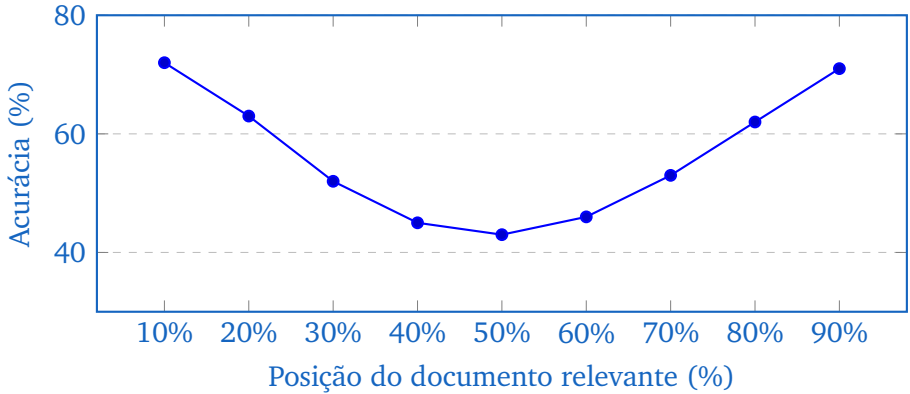
Liu, N. F. et al. “Lost in the Middle: How Language Models Use Long Contexts.” TACL 2024.[a]

[a] Nelson F Liu et al. “Lost in the middle: How language models use long contexts”. Em: TACL 12 (2024), pp. 157–173.

- Pergunta: onde colocar a informação relevante em um contexto longo?
- Tarefa: QA multi-documento, um documento contém a resposta
- Variou: (a) número de documentos, (b) posição do documento relevante
- Testou GPT-3.5-Turbo, GPT-4, Claude e modelos *open-source*
- **Todos os modelos exibem curva de desempenho em forma de U sobre a posição**

Lost in the Middle: Curva de Desempenho em U

Acurácia em QA multi-documento em função da posição do documento relevante:



Pico no início e fim (*primacy / recency*), **vale no meio**.

Redesenhado a partir de Liu et al., Lost in the Middle (TACL 2024)

Lost in the Middle: Por que Isso Acontece?

- **Efeito de primazia**: *tokens* próximos à posição 0 recebem alta atenção nas camadas iniciais
  - A máscara causal força todos os *tokens* a atender ao início da sequência
- **Efeito de recência**: os últimos *tokens* estão na "memória de trabalho" ativa da atenção
- ***Tokens* do meio, grande distância $|i-j|$ de ambas as extremidades**
  - Mesmo sem decaimento explícito de PE, padrões de atenção aprendidos tendem a ser locais
- **O alcance efetivo de atenção do modelo é menor que sua janela de contexto**
- **Nota:** GPT-4, Claude e LLaMA-2 usam RoPE ou variantes, o fenômeno **não** é resolvido trocando o PE. É um padrão aprendido de atenção, não uma propriedade do *encoding*.

Lost in the Middle: Implicações Práticas

- **Simplesmente aumentar o comprimento do contexto NÃO é suficiente**
  - Um modelo com contexto 32K pode ter desempenho pior que um de 4K se a resposta estiver no meio
- **A estratégia de posicionamento dos documentos importa enormemente**
  - Colocar o contexto mais importante no início ou no fim
  - RAG (*Retrieval-Augmented Generation*): buscar apenas o trecho relevante
- **Extensão de contexto (Parte 4) precisa ser combinada com dados de treino de contexto longo**

Needle in a Haystack: Configuração

Referência

Kamradt, G. “LLMTest_NeedleInAHaystack” (2023), *benchmark* da comunidade[a]

[a] Greg Kamradt. LLMTest_NeedleInAHaystack: Pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack. 2023.

- Teste de estresse sintético para recuperação em contextos longos
- ***Needle*:** fato curto e específico inserido em posição controlada
  - Ex: “`A melhor pizzaria de Redmond é a Giovanni's.`”
- ***Haystack*:** texto irrelevante e longo preenchendo o contexto
- ***Depth* %:** fração do contexto antes da *needle* (0% = início, 100% = fim)
- **Varredura:** comprimento do contexto (eixo x) $\times$ *depth* % (eixo y) $\rightarrow$ *heatmap* 2D

Needle in a Haystack: Como Ler o *Heatmap*

Eixos do *heatmap*:

- Eixo x: comprimento do contexto (1K → 128K)
- Eixo y: *depth* % da *needle*
- Cor: acurácia de recuperação

Padrões de falha:

- Faixa vermelha em contexto longo + meio → *Lost-in-the-Middle*
- Vermelho em todas as profundidades → limite rígido de contexto
- Verde em todo lugar → modelo ideal

Padrão ilustrativo (modelo fraco):

depth \ comp.	4K	16K	64K	128K
10% (início)	✓	✓	✓	✓
30%	✓	✓	~	×
50% (meio)	✓	~	×	×
70%	✓	✓	~	×
90% (fim)	✓	✓	✓	~

Needle in a Haystack: Resultados Típicos

Modelo / PE	Contexto	Meio a 64K	Padrão geral
GPT-4 / RoPE	128K	$\approx$ 90–100%	Quase perfeito
GPT-3.5 / RoPE	16K	50–70%	Falha em longo+meio
LLaMA-2 / RoPE	4K	$<$ 20% a 16K+	Colapsa além de L
Claude-2 / RoPE	100K	$\approx$ 80%	Maioria verde, lacunas no meio
Modelos fracos	varia	$<$ 50% meio	Zona de falha em U

- Modelos melhores (maiores, mais dados de treino de contexto longo) têm acurácia mais forte no meio
- Extensão de contexto (Parte 4) primariamente estende o eixo x (maior L)
- **Estender L sem corrigir o formato U não é suficiente para tarefas reais de contexto longo**

Valores aproximados de avaliações publicadas de Needle-in-a-Haystack

Conclusão: Janela de Contexto ≠ Compreensão

Uma janela de contexto mais ampla é **necessária** mas **não suficiente** para bom raciocínio em contextos longos.

Para realmente se beneficiar de uma janela de contexto longa, um modelo precisa de:

- Dados de treino com dependências de longo alcance (não apenas sequências longas)
- Padrões de atenção que recuperem informação do meio do contexto de forma confiável
- Possivelmente: estratégias de recuperação ou *chunking* para trazer o conteúdo certo

Métodos de extensão de contexto estendem a janela, não corrigem o formato U.
O campo está ativamente combinando extensão com melhor treino de contexto longo.

Visão Geral

O Problema da Extrapolação: Visão Técnica

Por que RoPE falha além de L_{train}?

- As frequências θ_k do RoPE são calibradas para que em L_{train} todos os ângulos de rotação sejam visitados
- Dims de alta frequência (*low-k*): ângulo $i \cdot \theta_k$ dá muitas voltas mesmo para L_{train}
- Dims de baixa frequência (*high-k*): ângulo $i \cdot \theta_k$ quase não se move
- **Além de L_{train}: dims de baixa freq. atingem ângulos NUNCA vistos → OOD**
- **Solução: reescalar índices de posição para manter o ângulo efetivo *in-distribution***

Ângulo efetivo

$\varphi(i,k) = i \cdot \theta_k$ Problema OOD: $\varphi(i,k) >$ max ângulo de treino quando $i > L_{\text{train}}$

Position Interpolation (PI)

Referência

Chen, S. et al. “Extending Context Window of LLMs via Positional Interpolation.” arXiv 2023.[a]

[a] Shouyuan Chen et al. “Extending context window of large language models via positional interpolation”. Em: arXiv preprint arXiv:2306.15595 (2023).

Correção mais simples: escalar os índices de posição para ficarem dentro de $[0, L_{\text{train}}]$.

Reescalonamento uniforme

$$i \leftarrow i \cdot (L_{\text{train}}/L_{\text{test}})$$

- Ângulo efetivo: $\varphi'(i,k) = i \cdot (L_{\text{train}}/L_{\text{test}}) \cdot \theta_k \rightarrow$ permanece na faixa de treino
- Requer curto *fine-tuning* (~1000 passos) (adaptar aos ângulos interpolados, OOD no treino)
- Alcança contexto de 8K–32K a partir de modelo treinado em 2K
- **Problema:** escalonamento uniforme comprime dims de alta freq. desnecessariamente (perde resolução fina)

YaRN: *Yet Another RoPE extensioN*

Referência

Peng, B. et al. "YaRN: Efficient Context Window Extension of Large Language Models." ICLR 2024.[a]

[a] Bowen Peng et al. "YaRN: Efficient context window extension of large language models". Em: ICLR. 2024.

- **Diferentes frequências do RoPE precisam de fatores de escala diferentes**
- Dims de alta freq.: já bem cobertas, manter como estão
- Dims de baixa freq.: muito esparsas, precisam de interpolação (como PI)
- Dims intermediárias: escalonamento parcial (*NTK-aware*, do *Neural Tangent Kernel*)
- **YaRN aplica escalonamento não-uniforme: escala diferente por dimensão**
- Corrige temperatura de atenção ($\sqrt{d} \to \sqrt{d} \cdot t$) para evitar supressão de *scores*
- **Alcança 128K a partir de 2K/4K com ~400 passos de *fine-tuning*; melhor retenção de contexto curto que PI**

LongRoPE: Artigo

Referência

Ding, Y. et al. “LongRoPE: Extending LLM Context Window Beyond 2 Million Tokens.” arXiv 2024.[a]

[a] Yiran Ding et al. “LongRoPE: Extending LLM context window beyond 2 million tokens”. Em: arXiv preprint arXiv:2402.13753 (2024).

- Alvo: estender contexto para 2 048K *tokens* (2M+) usando LLaMA-2 7B/13B
- Inovação: fatores de reescalonamento não-uniformes por dimensão via busca evolucionária
- *Fine-tuning* em dois estágios para alcançar 2M preservando qualidade em contexto curto
- **Em 2M *tokens*: ainda competitivo em *Needle-in-a-Haystack***
- Mostra que RoPE é extensível muito além de 100K com o reescalonamento correto

LongRoPE: Problema com Escalonamento Uniforme (PI)

Por que reescalonamento uniforme (PI) tem desempenho inferior em contextos muito longos?

Índice dim k	θ_k (freq.)	**Problema c/ escala uniforme**	**Tratamento ideal**
Low k (0–15)	≈ 1 (rápida)	Dá muitas voltas $\rightarrow$ OK	Manter inalterado (s=1)
Mid k (16–55)	≈ 0.1	Parcial $\rightarrow$ OOD leve	Escala pequena
High k (56–63)	≈ 0.01 (lenta)	OOD, problema principal	Escala grande

PI aplica a mesma escala a **todas** as dimensões. Sobre-comprime dims *low-k* e sub-comprime *mid-k*.

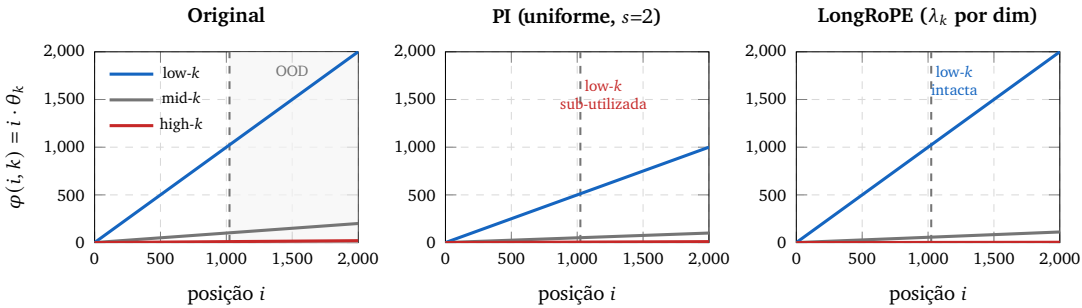
LongRoPE: reescalonamento por dimensão

$i \cdot \theta_k \leftarrow i \cdot \theta_k / \lambda_k$ $\quad$ λ_k específico por dimensão (busca evolucionária)

LongRoPE: Reescalonamento Não-Uniforme via Busca Evolucionária

- **Objetivo: encontrar fatores de escala por dimensão $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{d/2-1}]$**
- Função objetivo: minimizar perplexidade em L_{test} mantendo λ em faixa viável
- **Método: busca evolucionária (CMA-ES, *Covariance Matrix Adaptation Evolution Strategy*) sobre o vetor $\boldsymbol{\lambda}$**
- População de vetores $\boldsymbol{\lambda}$ candidatos avaliados por métrica de perplexidade
- Converge em poucas centenas de gerações, viável na prática
- **Resultado: dims *low-k* recebem $\lambda \approx 1$ (inalteradas); dims *high-k* recebem λ grande**
- O vetor $\boldsymbol{\lambda}$ final é fixo na inferência, zero custo em tempo de execução

LongRoPE: Original vs PI vs LongRoPE



PI trata todas as dimensões igualmente → compromete *low-k*.

LongRoPE aplica λ_k por dimensão → cada frequência recebe o tratamento certo.

LongRoPE: *Fine-Tuning* em Dois Estágios

Desafio: *fine-tuning* direto em 2M *tokens* é computacionalmente proibitivo.

Estágio	Contexto	Fatores λ	Descrição
Base	4K (LLaMA-2)	θ_k (inalterado)	Ponto de partida pré-treinado
Estágio 1	256K	$\lambda^{(1)}$ da busca	FT completo com dados de 256K
Estágio 2	2048K	$\lambda^{(2)}$ da busca	FT curto em 2M; ajustar λ

- Estágio 1 (256K): dados de *fine-tuning* completos; $\lambda^{(1)}$ otimizado para 256K
- Estágio 2 (2M): dados mínimos; $\lambda^{(2)}$ re-otimizado para alvo de 2M
- **Recuperação de contexto curto: λ separado para contextos curtos após estágio 2**
- Valores λ diferentes para inferência curta vs. longa $\rightarrow$ chaveamento dinâmico

Adaptado de Ding et al., LongRoPE (arXiv 2024)

LongRoPE: Resultados

Método	Ctx máx.	PPL em 2M	PPL 4K	NiH 1M meio
LLaMA-2 original	4K	$\gg$100 (falha)	5.5	$<$ 10%
PI	32K	$>$50 (degrada)	5.6	30–50%
YaRN	128K	$\sim$18	5.7	65–75%
LongRoPE	2048K	$\sim$9	5.6	$\approx$ 90%

- **LongRoPE alcança 2M com qualidade comparável em contexto curto**
- ***Needle-in-a-Haystack*: 90%+ de acurácia em 1M, *depth* 50%**
- Abordagem em dois estágios adiciona custo modesto de *fine-tuning* vs. treino completo em 2M

Dados de Ding et al., LongRoPE (arXiv 2024)

Métodos de Extensão de Contexto: Comparação

Método	Base PE	Tipo de escala	Ctx máx.	Custo de FT	Qual. ctx curto
PI	RoPE	Uniforme	32K–64K	Baixo (~1K passos)	Moderada
YaRN	RoPE	Não-uniforme (NTK)	128K	Baixo (~400 passos)	Boa
LongRoPE	RoPE	Por dim (evoluída)	2M+	Médio (2 estágios)	Excelente

- Todos os métodos fazem *fine-tune* de modelo RoPE existente, não treinam do zero
- *Trade-off*: extensão mais agressiva → mais *fine-tuning* e reescalonamento cuidadoso
- **Na prática: YaRN é o ponto ideal para a maioria dos casos (128K a baixo custo)**
- Questão aberta: combinar extensão com melhores dados de treino de contexto longo

Compilado de Chen et al. (2023), Peng et al. (2024), Ding et al. (2024)

Resumo

Método	Tipo	Ideia central	Extrapolação	Custo
PE Sinusoidal	Absoluto	Frequências sin/cos fixas	Pobre	Nenhum
Learned PE	Absoluto	Matriz de *embeddings* treinável	Nenhuma	$L \times d$ parâm.
RoPE	Relativo	Rotacionar Q, K por $i \cdot \theta_k$	Moderada	Tabela cos/sin
ALiBi	Relativo	*Bias* linear $-m \cdot \lvert i-j \rvert$	Boa	Desprezível
NoPE	Rel./Implíc.	Apenas máscara causal	Boa	Nenhum
PI	Extensão	Escalonamento uniforme	32K–64K	FT baixo
YaRN	Extensão	Escala não-uniforme NTK	128K	FT baixo
LongRoPE	Extensão	Escala por dim evoluída	2M+	FT médio

Referências I

[1] Shouyuan Chen et al. “Extending context window of large language models via positional interpolation”. Em: arXiv preprint arXiv:2306.15595 (2023).

[2] Yiran Ding et al. “LongRoPE: Extending LLM context window beyond 2 million tokens”. Em: arXiv preprint arXiv:2402.13753 (2024).

[3] Greg Kamradt. LLMTest_NeedleInAHaystack: Pressure testing LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack. 2023.

[4] Amirhossein Kazemnejad et al. “The impact of positional encoding on length generalization in transformers”. Em: NeurIPS. Vol. 36. 2023.

[5] Nelson F Liu et al. “Lost in the middle: How language models use long contexts”. Em: TACL 12 (2024), pp. 157–173.

[6] Bowen Peng et al. “YaRN: Efficient context window extension of large language models”. Em: ICLR. 2024.

[7] Ofir Press, Noah A Smith e Mike Lewis. “Train short, test long: Attention with linear biases enables input length extrapolation”. Em: ICLR. 2022.

[8] Jianlin Su et al. “RoFormer: Enhanced transformer with rotary position embedding”. Em: Neurocomputing 568 (2024), p. 127063.

[9] Ashish Vaswani et al. “Attention is all you need”. Em: NeurIPS. Vol. 30. 2017.