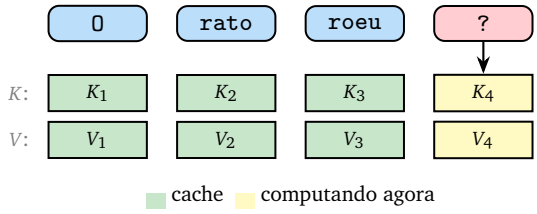
Instruções: Cada questão possui exatamente uma alternativa correta.

Questões

I) Durante a fase de **decode** de um decoder-only com KV-cache, o modelo gera o quarto token de uma sequência iniciada com o prompt de 3 tokens *"O rato roeu"*. O diagrama abaixo mostra o estado dos pares K,V nesse passo.



O que o diagrama ilustra sobre o funcionamento do KV-cache nesse passo?

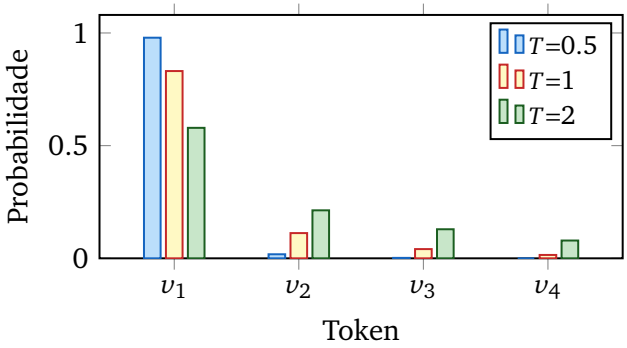
a) Apenas K_4 e V_4 são computados; K_1–K_3 e V_1–V_3 são reutilizados do cache.

b) Todos os K_1–K_4 e V_1–V_4 são recalculados integralmente a cada novo token gerado.

c) K_4 é computado, mas todos os V_1–V_4 são recalculados para combinar com a nova query Q_4.

d) O cache armazena apenas as queries; keys e values são sempre recalculados a partir dos tokens originais.

II) O gráfico abaixo mostra a distribuição de probabilidade sobre quatro tokens para os mesmos logits z=[3, 1, 0, −1], com três valores de temperatura.



Qual afirmação descreve corretamente o efeito da temperatura?

a) $T < 1$ achata a distribuição, aumentando a diversidade; $T > 1$ concentra a massa no token mais provável.

b) $T = 1$ produz distribuição uniforme; valores maiores concentram a massa progressivamente.

c) $T \to 0$ equivale ao argmax (greedy); $T > 1$ achata a distribuição, aumentando a diversidade da geração.

d) A temperatura não altera as razões de probabilidade entre tokens: se $P(v_i)/P(v_j) = r$ com T=1, essa razão permanece r para qualquer valor de T.

III) Ao avaliar um LLM, um pesquisador considera usar a **perplexidade** sobre um corpus de teste ou o benchmark **MMLU** (conhecimento enciclopédico em 57 domínios, formato de múltipla escolha). Qual afirmação descreve corretamente uma vantagem e uma limitação de cada métrica?

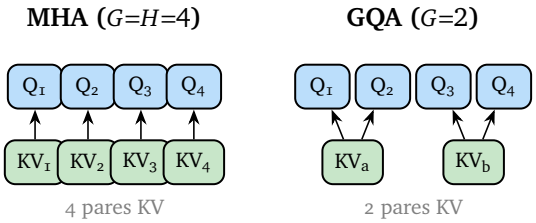
a) A perplexidade requer um conjunto de questões rotuladas, tornando-a mais custosa que o MMLU; o MMLU, por ser gerado automaticamente, é imune a contaminação do pré-treinamento.

b) A perplexidade independe do tokenizador, permitindo comparação direta entre modelos com vocabulários distintos; o MMLU mede exclusivamente raciocínio matemático em múltiplos passos.

c) A perplexidade é barata de computar e não exige dados rotulados, mas correlaciona-se imperfeitamente com o desempenho em tarefas específicas; o MMLU mede capacidades concretas, mas pode ser inflacionado por contaminação do corpus de pré-treinamento.

d) A perplexidade garante que modelos com valores menores terão melhor desempenho em qualquer tarefa downstream; benchmarks como o MMLU são imunes a esse problema por usarem questões de múltipla escolha.

IV) Considere um modelo com $H = 8$ cabeças de query, $d_k = 64$ e $L = 32$ camadas. O diagrama abaixo compara MHA e GQA com $G = 2$ grupos (usando H=4 na visualização):



Usando H=8, G=2, d_k=64, L=32: qual é o tamanho do KV-cache **por token** com GQA e quantas vezes menor é que o MHA?

a) 8 192 floats/token; 4× menor que MHA.

b) 8 192 floats/token; 8× menor que MHA.

c) 16 384 floats/token; 2× menor que MHA.

d) 4 096 floats/token; 8× menor que MHA.

V) Dado $\mathbf{x} = [3,\ 4]$ e $\boldsymbol{\gamma} = [1,\ 1]$ (ignorando ϵ), calcule RMSNorm($\mathbf{x}$) e identifique qual das opções corresponde ao resultado correto.

$$\mathrm{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\mathrm{RMS}(\mathbf{x})} \odot \boldsymbol{\gamma}, \qquad \mathrm{RMS}(\mathbf{x}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}.$$

a) $[\,-1.00,\ +1.00\,]$

b) $[\,0.849,\ 1.131\,]$

c) $[\,0.600,\ 0.800\,]$

d) $[\,0.857,\ 1.143\,]$

VI) Em um modelo Mixture-of-Experts (MoE) com $E = 4$ especialistas e $k = 1$ (top-1 routing), observa-se que, após o treinamento, o especialista 1 processa $\approx 97\%$ de todos os tokens, enquanto os especialistas 2–4 ficam essencialmente inativos. Qual fenômeno descreve essa situação e qual é a solução adotada?

a) **Overfitting** do router: o router memoriza os tokens de treino; solução: aumentar o número de especialistas E.

b) **Expert collapse**: ciclo de reforço positivo em que o especialista mais ativo recebe mais tokens, melhora mais e atrai ainda mais tokens; solução: perda auxiliar de balanceamento $\mathcal{L}_{\text{aux}} = \alpha\, E \sum_{i=1}^{E} f_i\, P_i$.

c) **Vanishing gradients**: gradientes desaparecem nos especialistas inativos; solução: conexões residuais no router.

d) **Mode collapse**: todos os especialistas convergem para a mesma função; solução: inicialização diversificada dos pesos.

VII) Na FFN SwiGLU, o cálculo é:

$$\text{FFN}(x) = \big(\text{SiLU}(xW_1) \odot xW_3\big)W_2,$$

enquanto a ReLU-FFN clássica usa $\text{FFN}(x) = \max(0,\, xW_1)\, W_2$. Qual é a principal vantagem do portão xW_3 em relação ao corte estático da ReLU?

a) A SwiGLU usa três matrizes em vez de duas, aumentando a capacidade bruta sem custo computacional adicional por token.

b) A SwiGLU aplica a não-linearidade duas vezes consecutivas (SiLU e depois W_2), aumentando a profundidade efetiva.

c) O portão xW_3 aprende dinamicamente quais dimensões do espaço oculto deixar passar, ao contrário da ReLU, que zera estaticamente valores negativos independentemente do contexto da entrada.

d) A SwiGLU elimina a necessidade de normalização por camada, pois o portão já controla a magnitude das ativações.

VIII) ☠️ No algoritmo de decodificação especulativa (Leviathan et al., 2023), o modelo alvo M_p e o modelo draft M_q atribuem as seguintes probabilidades a três tokens candidatos:

Token	p (alvo M_p)	q (draft M_q)
$\tilde{x}$	0.30	0.60
$\tilde{y}$	0.50	0.30
$\tilde{z}$	0.20	0.10

O draft propõe $\tilde{x}$. Qual é a probabilidade de aceitação de $\tilde{x}$ e, caso rejeitado, de qual distribuição é re-amostrado o token substituto?

a) Aceitação = 1.00 (sempre aceito, pois $p(\tilde{x}) > 0$); substituto amostrado de p sem modificação.

b) Aceitação = 0.50; substituto amostrado de $p'(x) \propto \max\big(0,\, p(x) - q(x)\big)$, que concentra a massa onde M_p supera M_q.

c) Aceitação = 0.50; substituto amostrado uniformemente sobre o vocabulário.

d) Aceitação = 0.30; substituto amostrado de q (draft).

IX) Uma equipe treina um modelo com $N = 8 \times 10^9$ parâmetros durante $D = 80 \times 10^9$ tokens e obtém perda de validação de 2.22. Satisfeitos com o resultado, param o treinamento. Em seguida, avaliam se vale a pena uma segunda rodada de treinamento prolongado até $D = 800 \times 10^9$ tokens. À luz das leis de escala do Chinchilla (Hoffmann et al., 2022), qual afirmação descreve corretamente a situação?

a) O modelo foi treinado de forma ótima ($D = 10N$ é a regra do Chinchilla); a segunda rodada não trará nenhuma melhoria pois a perda já saturou em 2.22.

b) O modelo foi sub-treinado em dados ($D^* = 20N = 160\text{B} > D$); a segunda rodada ainda reduzirá a perda, mas com *retornos decrescentes*.

c) O modelo foi super-treinado em dados ($D = 80\text{B} > D^* = N = 8\text{B}$); a segunda rodada causará sobreajuste e a perda de validação aumentará.

d) O modelo foi treinado de forma ótima ($D = 10N$); a segunda rodada melhorará a perda linearmente com os tokens adicionados.

X) Uma pesquisadora usa k-shot prompting com um LLM de janela de contexto $C = 2048$ tokens. O prompt de sistema ocupa 150 tokens, cada exemplar (pergunta + rótulo) ocupa 200 tokens e a consulta de teste ocupa 200 tokens. Com $k = 5$ o prompt cabe na janela. Ela quer testar $k = 10$. Qual afirmação está correta?

a) $k = 10$ é viável: $10 \times 200 = 2000 < 2048$, portanto há tokens suficientes.

b) $k = 10$ não é viável: o prompt totalizaria $150 + 10 \times 200 + 200 = 2350 > 2048$ tokens, excedendo a janela; o modelo precisaria truncar o contexto ou a geração.

c) $k = 10$ é equivalente a *fine-tuning* com 10 exemplos: em ambos os casos, o modelo atualiza sua distribuição de saída sem modificar os pesos.

d) $k = 10$ é viável, mas degrada o desempenho: mais exemplares no contexto causam *overfitting* aos rótulos, análogo ao fine-tuning com poucos dados.

XI) O **Multi-Query Attention** (MQA, Shazeer 2019) substitui os H pares K, V independentes do MHA por um **único** par K, V compartilhado entre todas as cabeças de query. Considere um modelo com $H = 8$ cabeças, $d_k = 64$ e $L = 32$ camadas. Qual é o tamanho do KV-cache **por token** com MQA e por quantas vezes é menor que o MHA equivalente?

a) 16 384 floats/token; 2× menor que MHA.

b) 4 096 floats/token; 8× menor que MHA.

c) 8 192 floats/token; 4× menor que MHA (mesmo resultado do GQA com G=2).

d) 4 096 floats/token; 4× menor que MHA.

XII) Num passo de decodificação, o modelo atribui as seguintes probabilidades (ordenadas de forma decrescente):

Token	P	P acumulada
`o`	0.45	0.45
`um`	0.30	0.75
`este`	0.15	0.90
`seu`	0.07	0.97
`...`	0.03	1.00

Com **top-p = 0.90** (nucleus sampling), qual é o conjunto *mínimo* de tokens que compõe o núcleo do qual o próximo token será amostrado?

a) Apenas `o` (probabilidade acumulada $= 0.45$).

b) `o`, `um` (probabilidade acumulada $= 0.75 < 0.90$).

c) `o`, `um`, `este` (probabilidade acumulada $= 0.90 \geq 0.90$).

d) `o`, `um`, `este`, `seu` (probabilidade acumulada $= 0.97 > 0.90$).

XIII) Um modelo de linguagem com vocabulário de 6 tokens $\mathcal{V} = \{$`o`, `gato`, `correu`, `leite`, `bebeu`, `fim`$\}$ atribui as seguintes probabilidades condicionais para cada posição da sequência *"o gato correu"*:

Token	$P(w_1)$	$P(w_2 \mid w_1)$	$P(w_3 \mid w_{1:2})$
`o`	**0.500**	0.050	0.050
`gato`	0.150	**0.250**	0.100
`correu`	0.150	0.300	**0.125**
`leite`	0.100	0.150	0.375
`bebeu`	0.050	0.150	0.250
`fim`	0.050	0.100	0.100

Os valores em **negrito** correspondem ao token observado em cada posição. Usando a definição

$$\mathrm{PPL}(w_1, \ldots, w_N) = \exp\left(-\frac{1}{N}\sum_{t=1}^{N} \ln P(w_t \mid w_{<t})\right),$$

qual é a **perplexidade** do modelo nessa sequência?

a) ≈ 3.4

b) 4

c) 8

d) 64

Soluções

I — Resposta: (a)

Durante a fase de *decode*, o KV-cache armazena os pares K_j, V_j de todos os tokens já processados. Para gerar o token na posição 4, o modelo computa K_4 e V_4 aplicando as projeções W^K e W^V ao embedding do novo token; os pares K_1, V_1, K_2, V_2 e K_3, V_3 são lidos diretamente do cache sem recálculo. Isso reduz o custo por passo de decodificação de $O(n^2 d)$ (sem cache) para $O(nd)$ (com cache), sendo essencial para geração eficiente em sequências longas.

II — Resposta: (c)

A temperatura escala os logits antes do softmax: $P_T(v) \propto \exp(z_v/T)$. Com $T \to 0$, apenas o maior logit contribui significativamente, equivalente ao arg max (greedy decoding). Com $T > 1$, os logits são reduzidos em magnitude, tornando a distribuição mais uniforme e aumentando a diversidade das amostras. O gráfico confirma: T=0.5 (azul) concentra ≈97.9% em v_1; T=2 (verde) distribui 57.9% em v_1 e o restante nos demais tokens. A opção (a) inverte os efeitos. A opção (b) confunde T=0 com distribuição uniforme. A opção (d) é incorreta pois a temperatura altera as probabilidades relativas: a razão $P_T(v_i)/P_T(v_j) = e^{(z_i - z_j)/T}$ depende de T.

III — Resposta: (c)

A perplexidade ($\text{PP} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i)\right)$) não exige anotação humana e é eficiente de calcular, sendo amplamente usada como proxy de qualidade durante o desenvolvimento e treinamento. Sua limitação principal é a correlação imperfeita com tarefas downstream: modelos com perplexidade similar podem diferir substancialmente em benchmarks como MMLU ou GSM8k. O MMLU (Hendrycks et al., 2021) avalia conhecimento em 57 domínios via múltipla escolha, fornecendo uma medida interpretável de capacidades específicas; porém, por ser público e amplamente divulgado, suas questões podem aparecer no corpus de pré-treinamento (*data contamination*), inflando artificialmente o desempenho reportado. A opção (a) inverte o requisito de dados rotulados: perplexidade não os exige, o MMLU sim. A opção (b) confunde dependência do tokenizador (perplexidade *depende* do tokenizador) e atribui ao MMLU o foco matemático do GSM8k. A opção (d) afirma falsamente que perplexidade baixa garante bom desempenho em qualquer tarefa downstream.

IV — Resposta: (a)

O tamanho do KV-cache por token é $2 \times G \times d_k \times L$ floats (2 tensores: K e V; G grupos KV; d_k dimensão por cabeça; L camadas):

$$\text{GQA } (G{=}2): \quad 2 \times 2 \times 64 \times 32 = 8\,192 \text{ floats/token.}$$

Com MHA ($G = H = 8$):

$$2 \times 8 \times 64 \times 32 = 32\,768 \text{ floats/token.}$$

Razão: 32 768/8 192 = 4×, não 8×, pois G=2 reduz H por um fator $H/G = 8/2 = 4$, não por $H = 8$. A opção (d) corresponderia ao MQA (G=1), onde há apenas 1 par KV compartilhado.

V — Resposta: (b)

$$\text{RMS}(\mathbf{x}) = \sqrt{\frac{3^2 + 4^2}{2}} = \sqrt{\frac{25}{2}} = \sqrt{12.5} \approx 3.536.$$

$$\text{RMSNorm}(\mathbf{x}) = \left[\frac{3}{3.536},\ \frac{4}{3.536}\right] \approx [0.849,\ 1.131].$$

A opção (a) é o resultado do *LayerNorm* ($\mu = 3.5$, $\sigma = 0.5$), que **subtrai a média** antes de normalizar; o RMSNorm não subtrai a média, por isso a segunda componente pode ser maior que 1. A opção (c) seria a normalização pela norma ℓ_2 ($\|\mathbf{x}\|_2 = 5 \Rightarrow [3/5,\ 4/5] = [0.6,\ 0.8]$); a opção (d) divide pela média $\mu = 3.5$.

VI — Resposta: (b)

O **expert collapse** é um ciclo de reforço positivo: o especialista mais capaz recebe mais tokens → acumula mais gradiente → torna-se ainda melhor → atrai ainda mais tokens. Os demais especialistas recebem gradiente próximo de zero e nunca aprendem. A solução padrão é acrescentar uma perda auxiliar de balanceamento à loss principal:

$$\mathcal{L}_{\text{aux}} = \alpha\, E \sum_{i=1}^{E} f_i\, P_i,$$

onde f_i é a fração de tokens roteados ao especialista i e P_i é o score médio do router para esse especialista. Minimizar $\mathcal{L}_{\text{aux}}$ incentiva distribuição uniforme entre especialistas (o Mixtral usa $\alpha = 0.01$).

VII — Resposta: (c)

Na ReLU-FFN, $\max(0, z) = 0$ sempre que $z < 0$, independentemente do token de entrada; o corte é *estático* e depende apenas do valor da ativação. Na SwiGLU, o portão xW_3 é um vetor computado a partir da entrada x: para cada token diferente, xW_3 produz valores distintos, de modo que o modelo *aprende* dinamicamente quais dimensões do espaço de dimensão d_{ff} suprimir ou amplificar. Isso confere maior expressividade sem alterar a complexidade assintótica. A opção (a) é parcialmente verdadeira (SwiGLU usa 3 matrizes), mas isso *aumenta* o custo computacional; a equivalência de parâmetros é mantida reduzindo d_{ff} para $\frac{8}{3}\, d_{\text{model}}$.

VIII — Resposta: (b)

Probabilidade de aceitação:

$$\min\left(1,\ \frac{p(\tilde{x})}{q(\tilde{x})}\right) = \min\left(1,\ \frac{0.30}{0.60}\right) = 0.50.$$

Distribuição de re-amostragem (se $\tilde{x}$ for rejeitado), $p'(x) \propto \max\big(0,\ p(x) - q(x)\big)$:

Token	p	q	$p - q$	p' (normalizado)
$\tilde{x}$	0.30	0.60	−0.30	0.00
$\tilde{y}$	0.50	0.30	+0.20	≈0.667
$\tilde{z}$	0.20	0.10	+0.10	≈0.333

p' concentra a massa nos tokens onde M_p supera M_q, garantindo que a distribuição final seja **idêntica** a amostrar diretamente de M_p, preservando a qualidade do modelo alvo por construção. A opção (a) confunde "$p > 0$" com "aceitar sempre"; apenas $p \geq q$ garante aceitação com probabilidade 1.

IX — Resposta: (b)

A regra prática do Chinchilla estabelece $D^* \approx 20N$. Para $N = 8 \times 10^9$ parâmetros:

$$D^* = 20 \times 8 \times 10^9 = 160 \times 10^9 \text{ tokens.}$$

O treinamento parou em $D = 80\text{B} < 160\text{B}$, portanto o modelo está **sub-treinado em dados**.

Continuar até $D = 800\text{B}$ ainda melhorará o modelo: a perda segue uma **lei de potência** $L(D) \propto D^{-\alpha_D}$ ($\alpha_D \approx 0.095$), de modo que ganhos existem além de D^*, mas são cada vez menores. Essa é a motivação por trás de modelos como o Llama 3 8B, treinado em ≈15T tokens ($\approx 1900 \times N^*$): do ponto de vista de *inferência*, vale gastar mais no treinamento para obter um modelo menor e mais barato de implantar.

As opções (a) e (d) erram a regra ($10N$ em vez de $20N$). A opção (a) agrava o erro ao afirmar que a perda satura exatamente em 20×, ela apenas *diminui mais lentamente*, mas não para. A opção (c) inverte a direção: com $D^* = 20N = 160\text{B}$, o modelo com $D = 80\text{B}$ está *abaixo*, não acima, do ótimo; além disso, LLMs não sofrem sobreajuste por excesso de dados, tipicamente quanto mais dados melhor.

X — Resposta: (b)

Com $k = 5$: $150 + 5 \times 200 + 200 = 1350 \leq 2048$ tokens (cabe). Com $k = 10$: $150 + 10 \times 200 + 200 = 2350 > 2048$ tokens (excede a janela em 302 tokens; o modelo precisaria truncar exemplares iniciais ou a geração, degradando a qualidade). A opção (a) esquece o prompt de sistema (150) e a consulta (200), computando apenas $k \times 200 = 2000 < 2048$ e concluindo erroneamente que cabe. A opção (c) confunde *in-context learning* com *fine-tuning*: o *few-shot prompting* não atualiza nenhum peso: o modelo usa os exemplares apenas como contexto durante a inferência. A opção (d) aplica indevidamente o conceito de *overfitting* ao ICL: como não há atualização de gradiente, não há sobreajuste no sentido clássico; mais exemplares tendem a melhorar o desempenho até a janela de contexto ser atingida.

XI — Resposta: (b)

Com MQA há apenas $G = 1$ par K, V, compartilhado pelas $H = 8$ cabeças de query. O tamanho do KV-cache por token é $2 \times G \times d_k \times L$:

$$\text{MQA } (G{=}1): \quad 2 \times 1 \times 64 \times 32 = 4\,096 \text{ floats/token.}$$

O MHA usa $G = H = 8$:

$$\text{MHA } (G{=}8): \quad 2 \times 8 \times 64 \times 32 = 32\,768 \text{ floats/token.}$$

Razão: 32 768 / 4 096 = 8×. O MQA é, portanto, o caso extremo da família GQA com G=1, oferecendo a maior redução de cache ao custo de uma pequena queda de qualidade (comparado ao GQA com G=2, que reduz 4× e preserva melhor a qualidade, ver questão IV).

XII — Resposta: (c)

O **nucleus sampling** (Holtzman et al., 2020) define o núcleo como o menor conjunto de tokens cuja probabilidade acumulada atinge ou supera o limiar p. Com $p = 0.90$:

- Após `o`: acum. = 0.45 < 0.90: insuficiente.
- Após `um`: acum. = 0.75 < 0.90: insuficiente.
- Após `este`: acum. = 0.90 ≥ 0.90: limiar atingido.

O núcleo é {`o`, `um`, `este`}; suas probabilidades são re-normalizadas e o próximo token é amostrado desse conjunto. A opção (d) inclui `seu` desnecessariamente, pois o limiar já foi satisfeito. A opção (b) é o top-p com p=0.75. A diferença-chave entre top-k e top-p é que o nucleus adapta o tamanho do conjunto ao *formato* da distribuição: se a distribuição for concentrada (e.g., $p_1 = 0.95$), o núcleo terá poucos tokens; se for difusa, terá muitos, comportamento mais robusto que um k fixo.

XIII — Resposta: (b)

As probabilidades dos tokens observados são $P(\texttt{o}) = \frac{1}{2}$, $P(\texttt{gato} \mid \texttt{o}) = \frac{1}{4}$ e $P(\texttt{correu} \mid \texttt{o}, \texttt{gato}) = \frac{1}{8}$. Aplicando a fórmula com $N = 3$:

$$\text{PPL} = \exp\left(-\frac{1}{3}\left(\ln\tfrac{1}{2} + \ln\tfrac{1}{4} + \ln\tfrac{1}{8}\right)\right) = \exp\left(\frac{1+2+3}{3}\,\ln 2\right) = \exp(2\ln 2) = 2^2 = 4.$$

Análise das alternativas incorretas:

- *(a)* ≈ 3.4: surge ao usar a média *aritmética* das probabilidades $(0.5 + 0.25 + 0.125)/3 \approx 0.292$ e tomar o seu inverso; a perplexidade equivale à média *geométrica* dos inversos, via $\exp(-\bar{\ell})$, não à média aritmética.
- *(c)* = 8: corresponde a $1/P(w_3) = 1/0.125 = 8$, considerando apenas o token de maior surpresa e ignorando as demais posições.
- *(d)* = 64: resultado de esquecer a normalização $1/N$: $\exp\left(-(\ln\frac{1}{2} + \ln\frac{1}{4} + \ln\frac{1}{8})\right) = 2^6 = 64$.