Aprendizado Profundo

Prova, Unidade 4

Prof. Lucas S. Kupssinskü

Data: _____

Nome: _____

Quadro de respostas (questões objetivas):

Questão	a	b	c	d	e	f	g	h
1								
2								
3								
4								
5								

Questões Objetivas (6.0 pts)

Cada questão possui apenas uma alternativa correta. Marque a letra escolhida no quadro de respostas.

1) (1.2) Considere um modelo *decoder-only* com $H = 8$ cabeças de *query*, dimensão por cabeça $d_k = 64$, $L = 32$ camadas e *batch size* 1. Você usa *KV-cache* durante a inferência. Quantos *floats* por token compõem o *KV-cache* para *Multi-Head Attention* (MHA, H cabeças de K e V) e para *Grouped-Query Attention* (GQA com $G = 2$ grupos para K e V), e qual a razão entre as duas?

   a) MHA: 32 768; GQA: 16 384; GQA é 2× menor que MHA.

   b) MHA: 32 768; GQA: 8 192; GQA é 4× menor que MHA.

   c) MHA: 16 384; GQA: 4 096; GQA é 4× menor que MHA.

   d) MHA: 32 768; GQA: 4 096; GQA é 8× menor que MHA.

   e) MHA: 32 768; GQA: 2 048; GQA é 16× menor que MHA.

   f) MHA: 16 384; GQA: 8 192; GQA é 2× menor que MHA.

   g) MHA: 65 536; GQA: 32 768; GQA é 2× menor que MHA.

   h) MHA e GQA armazenam o mesmo número de *floats* por token, pois ambos cacheiam todas as cabeças de *query*.

2) (1.2) Sobre as asserções abaixo a respeito de algoritmos de tokenização:

I. O **BPE** (*Byte-Pair Encoding*) inicia com um vocabulário de caracteres (ou *bytes*) e *cresce* o vocabulário fundindo iterativamente o par de tokens mais *frequente* no corpus de treinamento, até atingir o tamanho-alvo.

II. O **WordPiece** difere do BPE no critério de seleção: em vez do par mais frequente, mescla o par que *maximiza a verossimilhança* dos dados de treinamento; assim como o BPE, parte de um vocabulário pequeno e cresce o vocabulário a cada iteração.

III. A **tokenização em nível de caractere** (cada letra/caractere vira um token) é, em geral, preferível aos métodos sub-palavra (BPE/WordPiece) em LLMs modernos, pois produz um vocabulário muito menor, elimina por completo o problema de *out-of-vocabulary* e não acarreta custos adicionais relevantes de modelagem ou de tempo de treinamento.

a) Apenas I está correta.
b) Apenas II está correta.
c) Apenas III está correta.
d) Apenas I e II estão corretas.
e) Apenas I e III estão corretas.
f) Apenas II e III estão corretas.
g) Todas estão corretas.
h) Nenhuma está correta.

3) (1.2) Você está pré-treinando um *decoder-only* de 1B parâmetros com objetivo de *next-token prediction* sobre um corpus de texto. Após implementar a sua versão de *self-attention*, você observa que durante o treino o *loss* cai rapidamente para próximo de zero, mas, em inferência, o modelo é incapaz de gerar texto coerente, produzindo apenas ruído. Qual a explicação mais consistente **e** a melhor mitigação?

a) A taxa de aprendizado está alta demais, fazendo com que o *loss* caia rápido demais; reduzi-la em duas ordens de grandeza resolve a divergência inferência-treino.
b) O modelo está sofrendo *overfitting*; reduzir o número de parâmetros e adicionar *Dropout* resolve o problema.
c) A máscara causal não está sendo aplicada a $QK^\top$, então o modelo enxerga tokens futuros e aprende a copiá-los; aplicar $-\infty$ às entradas (i, j) com $j > i$ antes do *softmax* resolve.
d) O *tokenizer* está mal treinado, fragmentando palavras em caracteres individuais; treiná-lo com mais corpus resolve o problema.
e) Os *positional encodings* estão sendo somados à entrada; removê-los faz com que o modelo deixe de depender de informação posicional e generalize melhor.
f) O *softmax* da atenção precisa ser substituído por *sigmoid* ponto-a-ponto para problemas de geração autorregressiva.
g) O modelo precisa de *cross-attention*; *decoder-only* sem *cross-attention* é fundamentalmente incapaz de gerar texto coerente.
h) O problema é que se trata de *decoder-only*; trocar para *encoder-only* (tipo BERT) resolve, pois BERT é o estado da arte em geração de texto.

4) (1.2) Por que precisamos de *positional encoding* em *transformers*, enquanto RNNs não precisam dessa informação adicional?

a) Porque sem *positional encoding* o *softmax* saturaria e o gradiente desapareceria; o *positional encoding* ataca o problema de *vanishing gradient*.

b) Porque *self-attention* tem complexidade $O(n^2)$ e o *positional encoding* é o que reduz essa complexidade para aproximadamente $O(n)$.

c) Porque a função *softmax* usada na atenção é não-monótona, e o *positional encoding* linearizá-a.

d) Porque RNNs já incluem *positional encoding* implícito via *gates* de LSTM/GRU; *transformers* omitem esse *gating*.

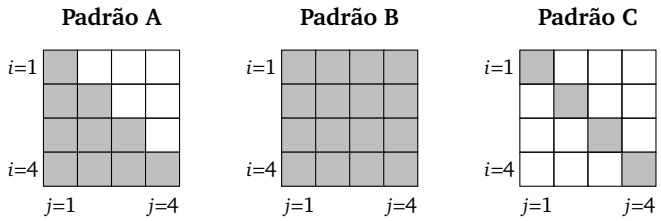
e) Porque *self-attention* é permutação-equivariante: sem PE, "o gato persegue o rato" e "o rato persegue o gato" produzem as mesmas representações. RNNs codificam posição implicitamente pela ordem da recursão.

f) Porque *positional encoding* é necessário apenas durante a inferência; em treino, o gradiente fornece a informação posicional implicitamente.

g) Porque *transformers* usam apenas multiplicações e somas (sem recursão), e *positional encoding* é a forma de introduzir não-linearidade na rede.

h) Porque *positional encoding* é uma técnica de regularização, equivalente em efeito ao *Dropout* aplicado à camada de *embeddings*.

5) (1.2) Os três padrões abaixo representam matrizes de atenção A_{ij} de uma sequência de 4 tokens. Células **cinzas** indicam pares (i, j) atendidos (peso possivelmente não-nulo após *softmax*); células **brancas** indicam pares mascarados ($-\infty$ antes do *softmax*, peso 0 depois).



Qual padrão é compatível com o pré-treinamento de um modelo *decoder-only* autorregressivo (GPT, Llama)?

a) Padrão A: atenção causal, necessária para *next-token prediction* sem ver tokens futuros.

b) Padrão A: ele permite que cada posição i veja apenas a si mesma, garantindo independência entre os tokens durante o treino.

c) Padrão B: *decoder-only* requer atenção bidirecional para construir representações contextualizadas dos tokens passados.

d) Padrão B: sem máscara o modelo aprende mais rapidamente; a propriedade autorregressiva é imposta apenas em inferência.

e) Padrão C: cada token só atende a si mesmo, o que evita vazamento de informação futura.

f) A e B são equivalentes do ponto de vista do treinamento autorregressivo.

g) A e C bloqueiam tokens futuros, então ambos são corretos para *decoder-only*.

h) Nenhum: *decoder-only* não usa atenção mascarada; apenas *cross-attention* é mascarada.

Questões Dissertativas (4.0 pts)

6) (2.0) Marque **Verdadeiro** ou **Falso** nas asserções abaixo. Em caso **Falso**, proponha uma modificação *não-trivial* que torna a asserção verdadeira (a simples negação **não** é uma modificação válida).

a) ( ) V ( ) F Em *Multi-Head Attention* com H cabeças e dimensão d_{model}, cada cabeça opera com dimensão $d_k = d_{\text{model}}/H$ e a saída final é a concatenação das H projeções, seguida por uma projeção linear W^O que recompõe a dimensão d_{model}.

b) ( ) V ( ) F O *positional encoding* sinusoidal proposto por Vaswani et al. (2017) introduz, para cada uma das d dimensões do *embedding*, um vetor de parâmetros treináveis aprendidos por descida de gradiente durante o treinamento, o que permite à rede adaptar a representação posicional ao corpus específico.

c) ( ) V ( ) F Modelos *encoder-only* (e.g., BERT) usam atenção bidirecional e são tipicamente pré-treinados com objetivo de *masked language modeling*; modelos *decoder-only* (e.g., GPT) usam atenção mascarada (causal) e são pré-treinados com objetivo de *next-token prediction*.

d) ( ) V ( ) F O algoritmo BPE inicia com um vocabulário pequeno (caracteres ou *bytes* individuais) e cresce iterativamente o vocabulário fundindo o par de tokens *menos* frequente, até atingir o tamanho-alvo do vocabulário.

e) ( ) V ( ) F *Multi-Query Attention* (MQA) usa H cabeças independentes para Q, K e V; portanto, reduzir o número de cabeças *de Q* é a forma de diminuir o tamanho do *KV-cache* durante a inferência.

7) (2.0) Considere uma camada de *self-attention* causal de **cabeça única** aplicada a uma sequência de 3 tokens (x_1, x_2, x_3). As matrizes Q, K, V já foram calculadas e têm dimensão 3×1 (sequência de comprimento 3, dimensão de chave $d_k = 1$):

$$Q = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad V = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}.$$

A operação aplicada é $O = \text{softmax}\big(\text{mask}(QK^\top)/\sqrt{d_k}\big)\,V$, onde $\text{mask}(\cdot)$ aplica a máscara causal.

a) (0.5) Compute a matriz de pontuações $S = QK^\top$.

b) (0.5) Aplique a máscara causal a S e indique a matriz S_{mask} resultante.

c) (0.5) Aplique *softmax* linha a linha em S_{mask} e indique a matriz de pesos de atenção A.

d) (0.5) Compute a saída $O = A \cdot V$. Apresente os três valores O_1, O_2, O_3.