Universidade de São Paulo
Instituto de Matemática e Estatística
Aluna: Nara Avila Moraes
Nro. USP: 5716734

2o. semestre, 2025
Prova 02 - 05/12

MAE5911/IME: Fundamentos de Estatística e Machine Learning.
Prof.: Alexandre Galvão Patriota

Questão 01: Descreva o mecanismo de atenção com múltiplas cabeças (Multi-head Attention). Apresente o desenvolvimento como feito em sala, considerando a versão "mascarada", para identificar médias ponderadas dinamicamente.

O mecanismo de atenção é o que transforma representação semântica de tokens em representação contextualizada, garantindo que os tokens sejam reinterpretados de acordo com o contexto em que é referido. O mecanismo consiste em durante o treino aprender as matrizes W_Q e W_K que representam parâmetros de *query*, o que este token procura, e *key*, o que este token oferece, tais que

$$Q_i = x_i * W_Q$$

$$K_i = x_i * W_K,$$

no qual, durante o teste, o embedding x de cada token é redefinido para um novo valor utilizando a fórmula

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + M\right) V \;=\; \sum_j A_{ij}\, V_j$$

na qual M é a máscara que apenas atribui contribuição não nula para tokens em relação a si mesmo e aos tokens predecessores, garantindo uma análise causal de contexto e V é a estatística suficiente sobre a qual tomamos a expectativa, também calculado via matriz de parâmetros W_V, tal que $V_i = x_i * W_V$. O resultado final é a matriz Attention, que tem dimensão $n \times d_{model}$, para n tokens e dimensão do embedding d_{model}, no modelo com uma cabeça.

Para o modelo com multiplas cabeças, o *Multi-head Attention*, a dimensão do embedding é dividida em h cabeças, cada uma com dimensão $d_k = d_{model}/h$. Cada cabeça aprende suas próprias matrizes de parâmetros $W_Q^{(i)}$, $W_K^{(i)}$ e $W_V^{(i)}$, para $i = 1, \ldots, h$. O mecanismo de atenção é aplicado separadamente para cada cabeça, resultando em h matrizes de atenção distintas.

Este modelo tem a vantagem de reduzir a dimensionalidade do embedding e permitir a paralelização do cálculo da matriz de atenção, além de permitir que cada cabeça foque em diferentes partes da sequência de entrada x, analisando separadamente diferentes aspectos de relacionamento entre tokens, como por exemplo, semântica, sintaxe, ou interdependências. Essas matrizes são então concatenadas e projetadas de volta para a dimensão original do embedding usando uma matriz de projeção de parâmetros adicional W_O, que combina as matrizes de atenção com pesos diferentes, adicionando uma camada adicional de refinamento do contexto, podendo adicionalmente ser utilizada para reduzir a dimensionalidade do embedding. A fórmula do modelo com multi-head attention é:

$$\text{MultiHead}(Q,K,V) = \text{Concat}\Big(\text{Attention}(QW_1^Q,\, KW_1^K,\, VW_1^V),\; \ldots,\; \text{Attention}(QW_h^Q,\, KW_h^K,\, VW_h^V)\Big)\, W^O.$$

Exemplo do cálculo de *multi-head attention*, partindo dos seguintes blocos W_Q, W_K e W_V de parâmetros de atenção treinados, e $W_O = I_4$ neste exemplo por simplicidade, para $h = 2$:

Head 1

$$W_Q^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, W_K^{(1)} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, W_V^{(1)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Head 2

$$W_Q^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, W_K^{(2)} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, W_V^{(2)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Considere um prompt com $n = 3$ tokens:

$$(\text{Life},\ \text{is},\ \text{awesome})$$

Cada token é representado por um embedding de dimensão $d_{\text{model}} = 4$. Para fins ilustrativos consideramos a seguinte matriz de embeddings:,

$$X = \begin{bmatrix} \text{Life} \\ \text{is} \\ \text{awesome} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 4}.$$

Cálculo das projecoes Q, K, V para cada token:

$$Q^{(i)} = XW_Q^{(i)}, \quad K^{(i)} = XW_K^{(i)}, \quad V^{(i)} = XW_V^{(i)},$$

Head 1

$$Q^{(1)} = \begin{bmatrix} 2 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, K^{(1)} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \\ 1 & 3 \end{bmatrix}, V^{(1)} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \\ 1 & 2 \end{bmatrix}$$

Head 2

$$Q^{(2)} = \begin{bmatrix} 0 & 2 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}, K^{(2)} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 3 & 1 \end{bmatrix}, V^{(2)} = \begin{bmatrix} 1 & 1 \\ 2 & 0 \\ 1 & 2 \end{bmatrix}$$

Calculamos agora, para $d_k = d_{\text{model}}/h = 2$:

$$S^{(i)} = \frac{Q^{(i)} {K^{(i)}}^\top}{\sqrt{d_k}}$$

Head 1

$$S^{(1)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 4 & 2 & 2 \\ 3 & 2 & 4 \\ 4 & 3 & 7 \end{bmatrix}$$

Head 2

$$S^{(2)} = \frac{1}{\sqrt{2}} \begin{bmatrix} 2 & 4 & 2 \\ 2 & 3 & 4 \\ 3 & 4 & 7 \end{bmatrix}$$

Aplicamos em seguida a máscara causal, o que impede que o token t seja co-relacionado com tokens em posições subsequêntes (futuras) $j > t$, exatamente igual ocorre no treinamento. Os scores mascarados são obtidos por:

$$\widetilde{S}^{(i)} = S^{(i)} + M, \qquad M = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix},$$

Note que a máscara zera as probabilidades das posições não causais:

$$\widetilde{S}^{(1)}_{1,:} = \big(s_{11}, -\infty, -\infty\big) \quad \Longrightarrow \quad \text{softmax}\big(\widetilde{S}^{(1)}_{1,:}\big) = \frac{\big(e^{s_{11}}, e^{-\infty}, e^{-\infty}\big)}{e^{s_{11}} + e^{-\infty} + e^{-\infty}} = (1, 0, 0),$$

Aplicando a função softmax linha a linha aos scores mascarados, obtemos as matrizes $A^{(1)}$ e $A^{(2)}$.

Head 1

$$A^{(1)} = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.670 & 0.330 & 0.000 \\ 0.102 & 0.050 & 0.848 \end{bmatrix}$$

Head 2

$$A^{(2)} = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.330 & 0.670 & 0.000 \\ 0.050 & 0.102 & 0.848 \end{bmatrix}$$

A saída da cabeça i é

$$\text{head}_i = A^{(i)} V^{(i)} \in \mathbb{R}^{3\times 2}.$$

Head 1

$$A^{(1)}V^{(1)} =\simeq \begin{bmatrix} 2.000 & 1.000 \\ 1.670 & 1.330 \\ 1.102 & 1.898 \end{bmatrix}$$

Head 2

$$A^{(2)}V^{(2)} =\simeq \begin{bmatrix} 1.000 & 1.000 \\ 1.670 & 0.330 \\ 1.102 & 1.747 \end{bmatrix}$$

Considerando, por simplicidade, a projeção de saída como a identidade,

$$W^O = I_4,$$

a saída final do bloco de *multi-head attention* é

$$\text{MultiHead}(Q, K, V) = \text{Concat}(H^{(1)}|H^{(2)})W_O \in \mathbb{R}^{3\times 4}.$$

Multi-Head Attention

$$MultiHead(Q, K, V) = \begin{bmatrix} 2.000 & 1.000 & 1.000 & 1.000 \\ 1.670 & 1.330 & 1.670 & 0.330 \\ 1.102 & 1.898 & 1.102 & 1.747 \end{bmatrix} \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 1.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$$

Resultado final:

Multi-Head Attention (novos *embeddings* contextualizados)

$$Attention(Q, K, V) = \begin{bmatrix} \text{Life} \\ \text{is} \\ \text{awesome} \end{bmatrix} = \begin{bmatrix} 2.000 & 1.000 & 1.000 & 1.000 \\ 1.670 & 1.330 & 1.670 & 0.330 \\ 1.102 & 1.898 & 1.102 & 1.747 \end{bmatrix}$$

```
# ------------------------------------------------------------------------------------------
# Cada camada N da rede neural receberá um módulo de atenção multi-cabeças definido da seguinte forma:
# ------------------------------------------------------------------------------------------
self$MM <- torch::nn_module_list(lapply(1:N_Layers,
  function(x) torch::nn_multihead_attention(
    n_embd,               # d_model: dimensão do vetores de embedding
    N_Head,               # h: número de cabeças
    dropout = p0,         # tecnica para reduzir o overfitting
    batch_first = TRUE))) # informa a disposição dos dados à biblioteca torch
```

Listing 1: Definição dos blocos de multi-head attention treináveis

```
# ------------------------------------------------------------------------------------------
# Cria uma matriz TxT, sendo T = número de tokens na sequência de entrada
# Garante que será tratado no mesmo device que X (CPU ou GPU) para que eles possam ser operados juntos
# Aplica 1 à diagonal superior, transforma 1 em booleano TRUE para que a biblioteca torch saiba quais posiçõ
    es precisam ser mascaradas
# ------------------------------------------------------------------------------------------
wei <- torch::torch_triu(
  torch::torch_ones(T, T, device = x$device),
  diagonal = 1)$to(dtype = torch::torch_bool())
```

Listing 2: Definição da Máscara causal

```
for (j in 1:self$N) {

  # ------------------------------------------------------------------------------------
  # 1) Pré-normalização (LayerNorm)
  #    Normalização dos embeddings, já que todos embeddings interagirão com todos os outros.
  #    A normalização não altera a informação e estabiliza o treinamento.
  # ------------------------------------------------------------------------------------
```

```
  QKV <- self$scale1[[j]](output)

  # ------------------------------------------------------------------------------------
  # 2) Chamada da função Multi-Head Self-Attention mascarada do torch que calcula Q, K e V internamente a
    partir de QKV.
  # ------------------------------------------------------------------------------------
  attn_out <- self$MM[[j]](
    query      = QKV,     # Input para o cálculo de Q
    key        = QKV,     # Input para o cálculo de K
    value      = QKV,     # Input para o cálculo de V
    attn_mask  = wei,     # Máscara causal
    need_weights = FALSE  # Dispensa o retorno dos pesos utilizados no cáculo da média
  )[[1]]                  # Guarda apenas o resultado da média ponderada, mas não os pesos

  # ------------------------------------------------------------------------------------
  # 3) Conexão residual após atenção
  # Mantém a informação original dos embeddings e evita que a transformação da atenção distorça
    excessivamente a representação.
  # Também estabiliza o fluxo do gradiente.
  # ------------------------------------------------------------------------------------
  output <- output + attn_out

  # ------------------------------------------------------------------------------------
  # 4) Feed-Forward Network + residual
  #  Aplicação de uma MLP ponto-a-ponto em cada token individualmente, capturando relações não lineares
    permitindo que a rede aprenda padrões complexos.
  #  Mantendo o resíduo e aplicando a normalização para estabilizar a operação.
  # ------------------------------------------------------------------------------------
  output <- output + self$FFN[[j]]( self$scale2[[j]](output) )
}
```

Listing 3: Aplicação da atenção multi-cabeças mascarada em cada camada

Questão 02: Treine um modelo de linguagem com dimensão embedding de 128, duas camadas e duas cabeças para os dados de Shakespeare.

Questão 03: Repita o item anterior sem o mecanismo de atenção (com múltiplas cabeças) e descreva o que ocorre no treinamento.
