I. Ví dụ về cách hoạt động của Encoder

Xét Encoder trong Transformer xử lý một câu ngắn: "Xin chào".

Thiết lập trước các tham số của Encoder:

* Kích thước Embedding (​):
* Số lượng Head trong Multi-Head Attention ():
* Kích thước mỗi Head ():
* Kích thước lớp ẩn trong Feed-Forward Network (​):
* Input: Câu "Xin chào" (2 token)

Các ma trận trọng số (​) và biases (​) trong ví dụ này sẽ đượcgiả định sẵn.

1. Bước 1: Input Embedding

Mỗi từ trong câu input được chuyển đổi thành một vector embedding. Giả sử chúng ta có embedding như sau (kích thước ):

Ma trận embedding đầu vào X ():

2. Bước 2: Positional Encoding (PE)

Công thức Positional Encoding:

Trong đó:

* : vị trí của từ trong câu (từ đến ).
* : chỉ số chiều (từ đến )

Chúng ta cần thêm thông tin vị trí vào các embedding. Sử dụng công thức sin/cos chuẩn với .

* Vị trí pos=0:
  + (dim ):
  + (dim ):
  + (dim ):
  + (dim ):
* Vị trí pos=1:
  + (dim):
  + (dim ):
  + (dim ):
  + (dim ):

Cộng Embedding và Positional Encoding:

Đây là ma trận input ​ () sẽ đi vào lớp Multi-Head Attention.

3. Bước 3: Multi-Head Self-Attention

Chúng ta có 2 head (), mỗi head có .

3.1 Tính toán Q, K, V cho mỗi Head:

Input là ​ (). Chúng ta cần các ma trận trọng số cho mỗi head. Các ma trận này sẽ chiếu input Xemb​ từ xuống .

* + Head 1: Giả sử các ma trận trọng số ():

​​

Tính :

* + Head 2: Giả sử các ma trận trọng số ():

​​ Tính :

3.2 Tính Scaled Dot-Product Attention cho mỗi Head:

Công thức:

* + Head 1:
    - Tính Scores:
    - Scale\_Scores: Chia cho
    - Softmax (theo hàng): Tính softmax cho từng hàng để có trọng số attention.
    - Tính Output Head 1:
  + Head 2: (Làm tương tự)
    - Tính Scores:
    - Scaled\_Scores
    - Softmax
    - Output Head 2

3.3 Concatenate các Head Output:

Nối ​ (2x2) và ​ (2x2) lại để được ma trận (shape 2x4).

3.4 Chiếu lên ma trận tuyến tính:

Nhân với ma trận trọng số (shape 4x4). Giả sử là ma trận đơn vị:

Output của Multi-Head Attention là .

4. Bước 4: Add & Norm (Sau Multi-Head Attention)

4.1 Add (Residual Connection):

Cộng output của MHA (​) với input ban đầu của lớp MHA (​).

4.2 Layer Normalization:

Chuẩn hóa từng vector (từng hàng của ​).

* + Hàng 1 (Vector ):
  + Hàng 2 (Vector ): (Tính tương tự)
  + Sau Layer Norm (giả sử scale và shift ):

5. Bước 5: Feed-Forward Network (FFN)

Lớp này áp dụng một mạng nơ-ron 2 lớp với input là ​ (shape 2x4).

Công thức:

​

* Lớp tuyến tính 1: Input () -> Hidden (). Cần ​ (shape ) và ​ (shape ).
* Kích hoạt ReLU:
* Lớp tuyến tính 2: Hidden () -> Output (). Cần ​ (shape ) và (shape ).

Giả sử ​ có giá trị như sau

* + ​
  + ​

Kết quả FFN (shape ):

6. Bước 6: Add & Norm (Sau FFN)

6.1 Add (Residual Connection):

Cộng output của FFN () với input của FFN (Xnorm1​).

6.2 Layer Normalization:

Chuẩn hóa từng hàng của ​.

* + Hàng 1:
  + Hàng 2:

Output cuối cùng của lớp Encoder này (sau Layer Norm, giả sử ):

II. Ví dụ về cách hoạt động của Decoder

Xét Decoder dịch "Xin chào" thành "Hello", sử dụng output của Encoder đã tính ở

Thiết lập tham số:

* Output Encoder (​): (Shape: 2x4)

* Đầu vào: <SOS>
* Tham số: .
* Tham số LayerNorm: Giả sử và .
* Từ vựng:

Bước A: Tạo Token thứ nhất ("Hello")

Đầu vào: Token <start>

* B1: Target Input Embedding: =
* B2: Target Positional Encoding (PE):
* B3: Masked Multi-Head Self-Attention:
  + Head 1:

Giả sử các ma trận trọng số

* + - (Sau mask và softmax)
  + Head 2:

Giả sử các ma trận trọng số

* + - (Sau mask và softmax)
  + Concatenate & Final Linear:
* B4: Add & Norm 1:
* B5: Multi-Head Cross-Attention:
  + Head 1:

Giả sử các ma trận trọng số

* + - (Từ X\_norm1\_dec)
    - (Từ E\_out)
    - (Từ E\_out)
    - (Softmax)
  + Head 2:

Giả sử các ma trận trọng số

* + - (Từ X\_norm1\_dec)
    - (Từ E\_out)
    - (Từ E\_out)
    - (Softmax)
  + Concatenate & Final Linear:
* B6: Add & Norm 2:
* B7: Feed-Forward Network (FFN):

Giả sử các ma trận tham số như sau

* B8: Add & Norm 3:
* B9: Final Linear & Softmax:

Giả sử các ma trận tham số như sau

* + Input:
  + (Shape 1x10).
* Logits cao nhất ở index 5
* Probabilities cao nhất ở index 5

Từ vựng:

* + Dự đoán: Token "Hello".

Bước B: Tạo Token thứ hai (<EOS>)

Thiết lập:

* Output Encoder (Shape: 2x4)
* Chuỗi đã tạo: <SOS> Hello
* Tham số: .
* Embeddings:
* Các ma trận trọng số và bias của Decoder: Sử dụng lại các trọng số đã giả định ở bước dự đoán "Hello".
* Tham số LayerNorm: Giả sử và .
* Từ vựng: (Mục tiêu là index 2).

Đầu vào: Chuỗi <SOS> Hello

* B1: Target Input Embedding:
* B2: Target Positional Encoding (PE):
* B3: Masked Multi-Head Self-Attention: (Xử lý cả <start> và "Hello")
  + Head 1:

Giả sử các ma trận tham số như sau

* + - Tính (Từ )
    - Tính (Từ )
    - Tính (Từ )

    - Áp dụng Mask :
    - (Softmax theo hàng)
  + Head 2:

Giả sử các ma trận trọng số

* + - Tính
    - Tính
    - V2 khoảng [[1.5, 2.0], [0.74, 2.31]]
    - khoảng [[2.25, 3.66], [2.37, 4.05]]
    - Áp dụng Mask :
  + Concatenate & Final Linear:

    - (Do là ma trận đơn vị)
* B4: Add & Norm 1:

* + Lấy vector cuối cùng (ứng với "Hello"):
* B5: Multi-Head Cross-Attention: (Chú ý đến , Query từ )
  + Head 1:

Giả sử các ma trận trọng số

* + - (Chiếu từ )
    - (Chiếu từ )
    - (Chiếu từ )
    - (Softmax)
  + Head 2:

Giả sử các ma trận trọng số

* + - (Chiếu từ last\_vec)
    - (Chiếu từ )
    - (Chiếu từ )
    - (Softmax)
  + Concatenate & Final Linear:
    - (Do là ma trận đơn vị)
* B6: Add & Norm 2:
  + (Sau LayerNorm)
* B7: Feed-Forward Network (FFN):

Giả sử các ma trận tham số như sau

* B8: Add & Norm 3:
  + (Sau LayerNorm)
* B9: Final Linear & Softmax:

Giả sử các ma trận tham số như sau

* + Input:
  + (Shape 1x10).
* Logit cao nhất ở index 2
  + Probabilities = softmax(Logits) (Shape 1x10).
* Xác suất cao nhất ở index 2

Từ vựng:

* + Dự đoán: Token <EOS>.