Name Date 15/08/2025 13:43

idx: batch of 4 : B
context size 8 : T

Target: 4x8 - just the next token for each token in the idx

Vocab size: 65 ⇒ number of possible characters
↓
V

Bigram : W - (65, 65) matrix

idx → (B X T X V)

→ logits ((B X T) x V) → 32 x 65

log counting table for 32 training data

→ max probability of Table (i, j)

i : sample i , j : correct index of target on that table

(Cross-entropy)

key is we don't have just 9 samples but actually 32

Self-attention:

output after embedding is

(4, 8, 32) → embedding dimensions

no of batch ↓ context length

First create a weight matrix that assigns the weights on previous observations using the lower triangular matrix created by applying softmax on either 0 or -inf initally

↦ initial weight matrix is a simple weighted average (TxT)

Now move on to the self-attention implementation:

Q: what the node is looking for

k: what am I containing.

head-size = H

$k = key(x)$: $B \times T \times H$

$Q = query(x)$: $B \times T \times H$

key, query: linear layer (C, H)

$W \Rightarrow Q \cdot k^T (-2, -1) : B \times T \times T$

keep the batch dimension

value: linear layer . $B \times T \times H$

$V = value(x)$: $B \times T \times H$

$out = W \cdot V : B \times T \times H$

value is like the value of this single head

. Attention is a communication mechanism. It can be seen as nodes in a directed graph looking at each other and aggregating info with a weighted sum from all nodes that point to them, with data dependent weights

. There is no notion of space. Attention simply acts over a set of vectors → we need positional encoding.

. Each example across batch dimension is processed independently

. with triangular masking → decoder which is used in autoregressive setting

We can also allow for all the nodes to

talk to eachother making it an encoder

- self-attention means Q, K, V all comes from the same source.
  cross-attention means we use different source
- scaled attention divides W by $\frac{1}{\sqrt{H}}$ → making W to be

Unit variance when Q, K are unit Variance → Softmax will stay diffuse and not saturate too much.

$W^Q : C \times H$ $W^T : C \times h$

$$Var(Q K^T) = H = \sum_{i=1}^{H} q_i k_i$$

$q_i, k_i \sim N(0,1)$

→ softmax are more spread out (diffuse) instead of sharpen.

Multi-head attention

Apply multiple attention in parallels.

usually: $\frac{n\_embd}{\text{no of heads}}$

This is similar to convolution

Transformers Architechture

· Multihead → FeedForward → Transform block

→ Repeat n-times.

Now the network is much deeper

→ We can improve with:

⊕ Residuals connections

adding the block input into output: $y = F(x)$

Make $y = F(x) + x$

→ make gradient flow backward easier

We also add a projection layer to ensure the output is always (B, T, C) (in case no.heads × head_size != C)

④ Layer norm

very similar to batch norm

just normalize row instead of column

For (B, T, C) → normalized across C instead of (B, T)

⑤ Apply Dropout.

Note:

our example: Vocabulary is character level

in practice: Sub-words is used (50,000

pre-train. just output text like our example → align with

QA docs → Train reward model (RM)