

? question @297  ★

stop following 71 views

Actions ▾

attention head implementation question

In the AttentionHead class in DLStudio, the implmentation includes

```
self.WQ = nn.Linear( max_seq_length * self.qkv_size, max_seq_length * self.qkv_size
)
...
Q = self.WQ( sent_embed_slice.reshape(sent_embed_slice.shape[0], -1).float() )
```

run code snippet

My understanding is that (ignoring batch axis) `sent_embed_slice` looks like $E = [e_1 e_2 \dots e_{N_w}]^T$ where $e_i \in \mathbf{R}^{s_{qkv}}$ and $E \in \mathbf{R}^{N_w \times s_{qkv}}$.

Then when we reshape, $\hat{E} = \text{reshape}(E) \in \mathbf{R}^{N_w \cdot s_{qkv}}$ and then multiply by $W_Q \in \mathbf{R}^{(N_w \cdot s_{qkv}) \times (N_w \cdot s_{qkv})}$. This represents the linear layer in the code above.

This W_Q could be written as a block matrix like $W_Q = \begin{bmatrix} W_{Q_1} \\ W_{Q_2} \\ \vdots \\ W_{Q_{N_w}} \end{bmatrix}$ where each $W_{Q_i} \in \mathbf{R}^{s_{qkv} \times (N_w \cdot s_{qkv})}$. The output after reshaping is is a $Q = \text{unreshape}(\hat{E}W_Q) \in \mathbf{R}^{N_w \times s_{qkv}}$

Each submatrix W_{Q_i} could be interpreted as a linear layer that takes in the concatenation of the partial word embeddings for the sequence and outputs a query embedding for the i th word. That means a unique W_{Q_i} matrix is being learned for each element in the sequence and it takes in all the elements in the sequence as its input.

This doesn't seem right however since my understanding was that a single W_Q matrix was learned for all elements of the embedding and that the transformer was sequence length invariant, unlike here where it very much depends on the sequence length.


I would think it would look something like $EW_Q = Q$ where $W_Q \in \mathbf{R}^{s_{qkv} \times s_{qkv}}$ and $Q \in \mathbf{R}^{N_w \times s_{qkv}}$.

other

~ An instructor (Fangda Li) endorsed this question ~

Edit

good question | 1

Updated 1 month ago by Moiz Rasheed 



the students' answer, where students collectively construct a single answer

[Click to start off the wiki answer](#)**the instructors' answer**, *where instructors collectively construct a single answer*

Here is Prof. Kak's response:

"In response to your comment, note that my transformer implementation lets the network decide for itself how much independence it wants to maintain in the learning of the different "rows" of the Q, K, and V tensors. [I am using "row" here to mean the representation learned for each word.] Instead of forcing the network to think of each word in an input sentence as a separate and independent entity, I am letting the network figure out on its own if that's what would get the best overall result. In highly idiomatic sentences, it is not uncommon for small groups of adjoining words to belong together and, during transduction, you must translate them together in one go. Perhaps my implementation makes it easier for the transformer to engage in that kind of learning.

I would have a similar answer to the Piazza post by the student. Theoretically, one would want the "tokens" for the words to be learned separately. My transformer implementation does NOT forbid that.

Nonetheless, the student has made a very good point --- a point worth talking about. It is also possible that perhaps I would get better results if I made the learning of the query, key, and value vectors for the words more independent."

[undo thanks](#) | 1

Updated 1 month ago by Fangda Li

followup discussions, *for lingering questions and comments*

Start a new followup discussion