Transformer Encoder Model for multiclass classification

Abdulsalam Bande

Model representation

- The model is the same model used as the encoder in the transformer network
- This new model as an extra Linear layer that outputs 6 features
- The 6 features go into a softmax to be converted to probabilities

Explaining Encoder Attention

1. The encoder first takes inputs as embeddings
2. These embeddings are then added to a positional encoding, The positional encoding have information on positions of sequences .

Explaining Encoder Attention

3. Next , Lets replicate these embeddings into 3 .
    a. We create 3 Linear Layers
    b. The 3 replicated embeddings all go into one of the 3 Linear Layers.
    c. The output of the 3 Linear Layers go through an attention formula (See below).
    d. Let us call the Outputs Value, Key and Query (V, K,Q) respectively
    e. First, the dot product of the Queries and the Keys is calculated.
    f. The result of the dot product gives information for every word , the attention to pay to other words
    g. The result of the dot product is divided by the dim of the embedding for scaling.
    h. A softmax is taken to convert attention to probabilities
    i. Next , the V IS multiplied with the previous result
    j. This V tensor, has the same dimension with the original input, only that it contains a context information

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

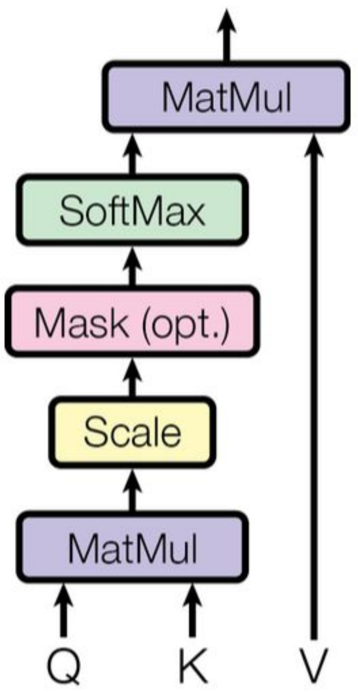
Encoder Attention Details

- In step 3b , we had just a set of 3 Linear Layers.
- We can improve by have multiple sets of these 3 Layers. And these sets are called heads.
- If we have multiple heads, the embedding dim can be separated to multiple Dimensions for each of the 3 Linear Layers.
- Example, if the original input is [3,12] . 3 here is the sequence length and 12 is the embedding dimension. Usually when we have 1 head(1 set of 3 Linear Layers), each Linear layer takes [3,12].
- But when we have 2 heads. The first [3,6] can go into head 1(for all the 3 Linear Layers) and the second [3,6] into head 2(for all the 3 Linear Layers).

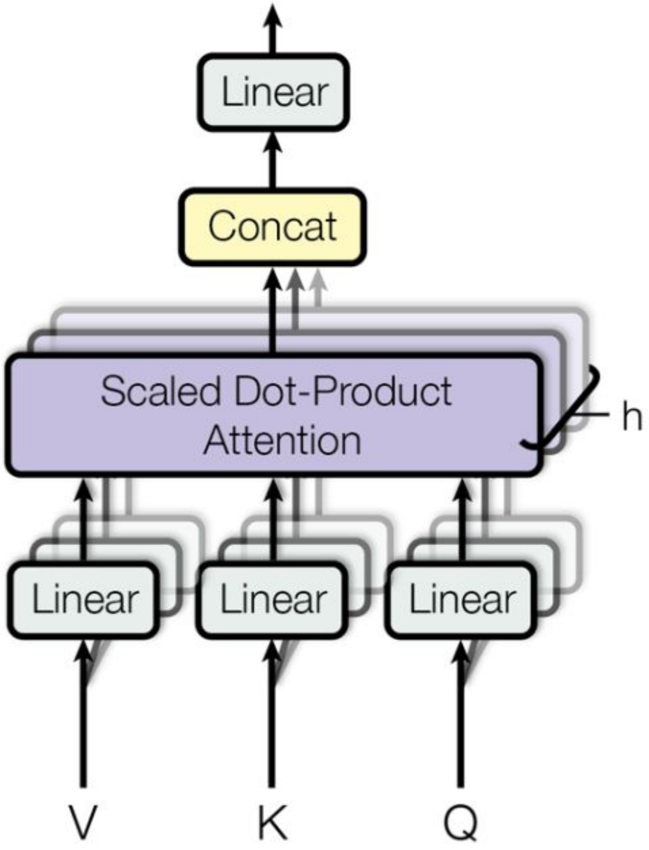
Encoder Attention Details

- The output of each head goes through the Scaled Dot-Product Attention formula (SDPA).
- SDPA of each head is concatenated.
- The concatenated tensor goes through other layers.
- Next Slides shows the diagram of the Scaled Dot-Product Attention and Multi-Head Attention

Scaled Dot-Product Attention



Multi-Head Attention



How do you associate Labels to attention?

- Let's say we have 6 Classes. And after the encoder, you have 6 probabilities for each class like [0.9, 0.1, 0.2, 0.5, 0.7, 0.3]
- Next, we can mask the values less than 0.5. So the tensor becomes [0.9, 0, 0, 0.5, 0.7, 0]

Below is what I propose

- Let's say the attention weights tensor has dimension [1,3]. 1 is the batch size and 3 is the sequence length (number of words). Let's say weights is [0.12,0.41,0.22]
- For each of the output probability >= 0.5 multiply it by the attention weight tensor.
- For 0.9 , the result is [0.108, 0.369, 0.198]. The max is 0.198
- So we can say for class0 (0.9 is at index 0) , an attentive words is the word at index 2 of the attention weights tensor.
- Note : There are many types of attention mechanisms.

Important detail

- In the context of Machine Translation. The weights of multi headed attention is usually of dimension (N, L, S). where N is the batch size, L is the target sequence length, S is the source sequence length.
- So the weight just tells you for each source word, how much attention is on the target words NOT how much attention is on the output of the Linear Layer.

Some points I missed before

- We can apply recall, precision and confusion matrix to access out model.
- We can also apply different feature selection techniques.