Transformers PseudoCode for Donut

March 6, 2024

1 Donut Encoder Pseudocode

Algorithm 1: Pseudocode for Donut's SwinTransformer Encoder

Input: $x \in \mathbb{R}^{C \times H \times W}$, a tensor representing the input image
Output: A set of embeddings $\{z_i | z_i \in \mathbb{R}^d, 1 \leq i \leq n\}$, where n is the number of patches and d is the dimension of the embeddings
Hyperparameters: P, patch size,
L, list of number of layers in each stage,
H, list of number of heads in each layer,
W, window size.
Parameters: W_{embed}, parameters for patch embedding,
W_{norm}, parameters for normalization layer (omitted in Donut),
W_{trans}, parameters of Swin Transformer layers.

$Z \leftarrow \text{ImageToTensor}(x)$; `// Converts image to tensor`
$Z \leftarrow \text{PatchEmbedding}(Z, W_{\text{embed}}, P)$; `// Maps image patches to embeddings`
$Z \leftarrow \text{DropPositional}(Z)$; `// Applies dropout after adding positional encodings`
foreach *stage in SwinTransformer* **do**
  for $l = 1$ **to** $L[stage]$ **do**
    $Z \leftarrow \text{LayerNorm}(Z)$; `// Applies layer normalization`
    $Z \leftarrow \text{WindowBasedMSA}(Z, W, H[\text{stage}])$; `// Window based multi-head self-attention`
    $Z \leftarrow \text{LayerNorm}(Z)$; `// Applies layer normalization again`
    $Z \leftarrow \text{MLP}(Z)$; `// Applies MLP block`
    if *stage* $<$ *NumberOfStages* **then**
      $Z \leftarrow \text{PatchMerging}(Z)$; `// Downsamples feature map for next stage`
    end
  end
end

2 Donut Decoder Pseudocode

Algorithm 2: Pseudocode for Donut's BART-based Decoder

Input: $z \in \mathbb{R}^{n \times d}$, a set of embeddings from the encoder
Output: A sequence of token probabilities $P \in \mathbb{R}^{m \times v}$, where each row represents a distribution over the vocabulary for the corresponding token
Hyperparameters: m, maximum sequence length,
v, size of token vocabulary,
L, number of decoder layers,
d, dimension of latent vectors.
Parameters: W_e, embedding matrix for token embeddings,
W_p, positional embedding matrix,
$W_{\text{layer\_norm}}$, parameters for layer normalization,
W_{MHAtt}, multi-head attention parameters,
W_{MLP}, parameters for the feed-forward network,
$W_{\text{lm\_head}}$, parameters for the language model head.

```
Initialize token sequence Y = [ ]
for i = 1 to m do
    Y_embed ← W_e Y + W_p ;        // Embed tokens and add positional encoding
    Y ← LayerNorm(Y_embed, W_layer_norm) ;        // Apply layer normalization
    for l = 1 to L do
        Y ← MultiHeadAttention(Y, z, W_MHAtt) ;   // Self-attention over the sequence
        Y ← LayerNorm(Y, W_layer_norm) ;          // Apply layer normalization
        Y ← MLP(Y, W_MLP) ;                       // Feed-forward network
    end
    P_i ← Softmax(W_lm_head Y) ;   // Project to vocabulary and apply softmax
    Y ← Append(Y, ArgMax(P_i)) ;   // Append the most probable next token
end
```