TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

The Transformer Part I

The Transformer

Attention is All You Need, Vaswani et al., June 2017

The Transformer has now essentially replaced RNNs in natural language applications.

The recent progress on the natural language understanding evaluation (GLUE) is based on general language modeling using Transformers.

Vector Sequences

Each layer in the Transformer has shape $L[T, J]$ where t ranges over the position in the input sequence and j ranges over neurons at that position (and omitting the batch index).

This is the same shape as layers in an RNN — a sequence of vectors $L[t, J]$.

When processing sentences, T is the sentence length.

Parallel Layer Computation

However, in the Transformer we can compute the layer $L_{\ell+1}[T, J]$ from $L_\ell[T, J]$ in $O(\ln(TI))$ time (in parallel).

The log comes from the time to compute a large sum in parallel using a tree of additions.

This is an important difference from RNNs which compute sequentially over time.

In this respect the Transformer is more similar to a CNN than to an RNN.

Self-Attention

The fundamental innovation of the Transformer is the self-attention layer.

For each position t in the sequence we compute an attention over the other positions in the sequence.

These attentions can be viewed as a soft directed graph over the positions where the attention from t_1 to t_2 can be viewed as a weighted directed edge from position t_1 to position t_2.

There is an intuitive analogy between this soft graph and a dependency parse tree.

Transformer Heads

In a dependency parse edges are typically labeled with grammatical roles such as “subject-of” or “object-of”.

The self attention layers of the Transformer we have “heads” which can be viewed as labels for dependency edges.

Self attention constructs a tensor $\alpha[k, t_1, t_2]$ — the strength of the attention weight (edge weight) from t_1 to t_2 with head (label) k.

Query-Key Attention

For each head k and position t we compute a key vector and a query vector with dimension I typically smaller than dimension J.

$$\begin{aligned}
\mathrm{Query}_{\ell+1}[k,t,i] &= W^Q_{\ell+1}[k,i,J]L_\ell[t,J] \\
\mathrm{Key}_{\ell+1}[k,t,i] &= W^K_{\ell+1}[k,i,J]L_\ell[t,J] \\
\alpha_{\ell+1}[k,t_1,t_2] &= \operatorname*{softmax}_{t_2} \frac{1}{\sqrt{I}}\, \mathrm{Query}_{\ell+1}[k,t_1,I]\mathrm{Key}_{\ell+1}[k,t_2,I]
\end{aligned}$$

Computing the Output

$$\begin{aligned}
\mathrm{Value}_{\ell+1}[k,t,i] &= W^{V}_{\ell+1}[k,i,J]L_{\ell}[t,J] \\
h^{1}_{\ell+1}[k,t,i] &= \alpha[k,t,T]\mathrm{Value}[k,T,i] \\
h^{2}_{\ell+1}[t,C] &= \tilde{h}_{\ell+1}[0,t,I];\cdots;\tilde{h}_{\ell+1}[K-1,t,I] \\
L_{\ell+1}[t,j] &= W^{0}[j,C]h^{2}[t,C]
\end{aligned}$$

Here semicolon denotes vector concatenation.

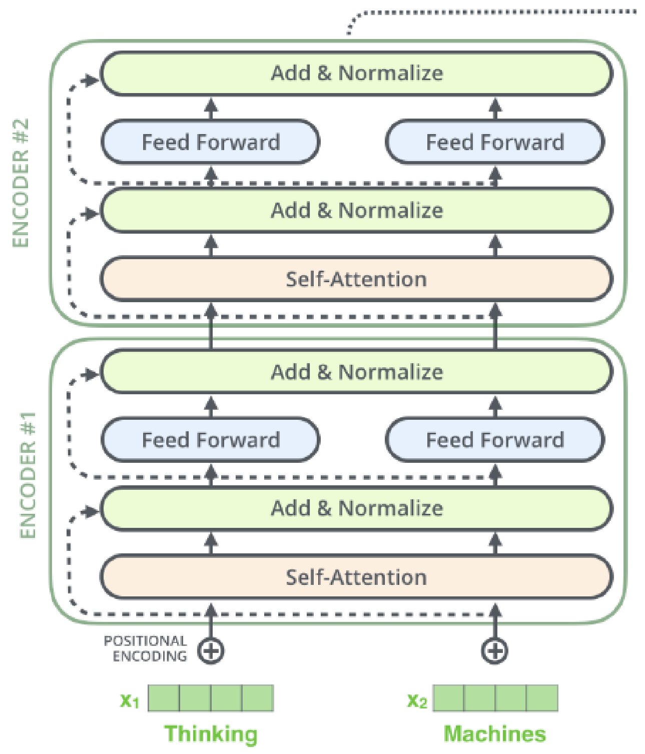
The Transformer Layer

Each "Transformer layer" consists of six "sublayers".

$$\begin{aligned}
\text{Self Attention: } L_{\ell+1}[T,J] &= \text{Self}(L_\ell[T,J]) \\
\text{Residual: } L_{\ell+2}[T,J] &= L_{\ell+1}[T,J] + L_\ell[T,J] \\
\text{Normalization: } L_{\ell+3}[T,J] &= \text{Norm}(L_{\ell+2}[T,J]) \\
\text{Feed Forward: } L_{\ell+4}[T,J] &= \text{FF}(L_{\ell+3}[T,J]) \\
\text{Residual: } L_{\ell+5}[T,J] &= L_{\ell+4}[T,J] + L_{\ell+3}[T,J] \\
\text{Normaliztion: } L_{\ell+6}[T,J] &= \text{Norm}(L_{\ell+5}[T,J])
\end{aligned}$$

The normalization layers use "layer normalization" — an alternative to batch normalization described in thenext unit.

Transformer Layers



Jay Alammar's blog

END