Multi-Head Self Attention

- Multiple "heads" analogous to different convolutional filters
- Let $E = [\text{sentence length, embedding dimension}]$ be the input sentence. This will be passed through three different linear layers to produce three mats:
  - Query $Q = EW^Q$: these are like the decoder hidden state in attention
  - Keys $K = EW^K$: these control what gets attended to, along with the query
  - Values $V = EW^V$: these vectors get summed up to form the output

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

Vaswani et al. (2017)
Self-Attention

Alammar, The Illustrated Transformer

sent len x sent len (attn for each word to each other)

softmax \left( \frac{Q K^T}{\sqrt{d_k}} \right)

sent len x hidden dim
Z is a weighted combination of V rows

Multi-head Self-Attention

Just duplicate the whole computation with different weights:

Mul-head Self-Attention

Properes of Self-Attention

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity per Layer</th>
<th>Sequential Operations</th>
<th>Maximum Path Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>O(n^d \cdot d)</td>
<td>O(1)</td>
<td>O(1)</td>
</tr>
<tr>
<td>Recurrent</td>
<td>O(n \cdot d^2)</td>
<td>O(n)</td>
<td>O(n)</td>
</tr>
<tr>
<td>Convolutional</td>
<td>O(k \cdot n \cdot d^2)</td>
<td>O(1)</td>
<td>O(log_2(n))</td>
</tr>
<tr>
<td>Self-Attention (restricted)</td>
<td>O(r \cdot n \cdot d)</td>
<td>O(1)</td>
<td>O(n/r)</td>
</tr>
</tbody>
</table>

- n = sentence length, d = hidden dim, k = kernel size, r = restricted neighborhood size
- Quadratic complexity, but O(1) sequential operations (not linear like in RNNs) and O(1) “path” for words to inform each other

Vaswani et al. (2017)
Transformers

Dimensions

- Vectors: \(d_{\text{model}}\)
- Queries/keys: \(d_k\), always smaller than \(d_{\text{model}}\)
- Values: separate dimension \(d_v\), output is multiplied by \(W^\circ\) which is \(d_v \times d_{\text{model}}\) so we can get back to \(d_{\text{model}}\) before the residual
- FFN can explode the dimension with \(W_1\) and collapse it back with \(W_2\)

\[
\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2
\]

Vaswani et al. (2017)

Architecture

- Alternate multi-head self-attention with feedforward layers that operate over each word individually

\[
\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2
\]

- These feedforward layers are where most of the parameters are
- Residual connections in the model: input of a layer is added to its output
- Layer normalization: controls the scale of different layers in very deep networks (not needed in A4)

<table>
<thead>
<tr>
<th>Model Name</th>
<th>(n_{\text{params}})</th>
<th>(n_{\text{layers}})</th>
<th>(d_{\text{model}})</th>
<th>(n_{\text{heads}})</th>
<th>(d_{\text{head}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-3 Small</td>
<td>125M</td>
<td>12</td>
<td>768</td>
<td>12</td>
<td>64</td>
</tr>
<tr>
<td>GPT-3 Medium</td>
<td>350M</td>
<td>24</td>
<td>1024</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>GPT-3 Large</td>
<td>760M</td>
<td>24</td>
<td>1536</td>
<td>16</td>
<td>96</td>
</tr>
<tr>
<td>GPT-3 XL</td>
<td>1.3B</td>
<td>24</td>
<td>2048</td>
<td>24</td>
<td>128</td>
</tr>
<tr>
<td>GPT-3 2.7B</td>
<td>2.7B</td>
<td>32</td>
<td>2560</td>
<td>32</td>
<td>80</td>
</tr>
<tr>
<td>GPT-3 6.7B</td>
<td>6.7B</td>
<td>32</td>
<td>4096</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>GPT-3 13B</td>
<td>13.0B</td>
<td>40</td>
<td>5140</td>
<td>40</td>
<td>128</td>
</tr>
<tr>
<td>GPT-3 175B or “GPT-3”</td>
<td>175.0B</td>
<td>96</td>
<td>12288</td>
<td>96</td>
<td>128</td>
</tr>
</tbody>
</table>

From Vaswani et al.

From GPT-3; \(d_{\text{head}}\) is our \(d_k\)
Optims setup	|	FLOPs /
update	|	% FLOPs MHA	|	% FLOPs FFN	|	% FLOPs attn	|	% FLOPs logit
---
OPT setups	| 4.3E+15	| 35%
2.7B	| 2.5E+16	| 29%
6.7B	| 1.1E+17	| 24%
13B	| 4.1E+17	| 22%
30B	| 9.0E+17	| 20%
66B	| 9.5E+17	| 18%
175B	| 2.4E+18	| 17%

Credit: Stephen Roller on Twitter

The ballerina is very excited that she will dance in the show.

- If this is in a longer context, we want words to attend *locally*
- But transformers have *no notion of position* by default

- Encode each sequence position as an integer, add it to the word embedding vector
- Why does this work?
- Original Transformer paper presents an encoder-decoder model

- Right now we don’t need to think about both of these parts — will return in the context of MT

- Can turn the encoder into a decoder-only model through use of a triangular causal attention mask (only allow attention to previous tokens)

Vaswani et al. (2017)