Transformers #1: Introduction to Transformer

RPL CV/DL reading group

Sebastian Bujwid

23 Feb 2021
Transformers theme: schedule

- **23 Feb** (today) (Sebastian): Introduction to the Transformer model
- **9 Mar** (Federico): Transformer models for Image
- (tentative) **23 Mar** (Yonk): Transformer models in different applications/domains
- (tentative) **6 Apr** (Sofia): Alternative approaches to Transformer
Agenda

1. Taxonomy: attention, self-attention, Transformer?
   - Why attention?
   - Why Transformer?
   - Discussion!

2. Transformer
   - Transformer vs. RNN; Transformer in general
   - Some specifics for text
   - Discussion!

3. Some details about the Transformer & results

4. Discussion!
Taxonomy
Attention! - definition

"Attention is a technique that mimics cognitive attention" (Wikipedia)

"(Your) Attention is (our) profit" (Instagram) ▲ Official sources might deny it.

"The ability to focus on one thing and ignore others" (Alex Graves, 2020) - I might have taken it out of context :/

"Looking at some places more than at others" (Bujwid, 2021)
Why attention (in neural networks)?
Why attention (in neural networks)?
Why attention (in neural networks)?

Anna är orolig för miljön så hon använder sin cykel året runt

(...)

Anna is concerned about the environment so she uses her bike all year round
Taxonomy

Attention mechanism

Specific models (architectures)
- Transformer
- BERT
- AlphaFold 2
- Vision Transformer
- iGPT
- (…)

Compatibility (similarity) function
- dot-product
- scaled dot-product
- cosine similarity
- additive
- (…)

How to look?
- hard attention
- soft attention
- (…)

What to look at?
- self-attention (intra-attention)
- cross-modal attention
- seq. 1 & seq. 2
- (…)

Type/domain of input
- text
- image
- video
- graph
- time series
- (…)
Why Transformer [1]?

Let's look at the issues with RNNs first!

**Issue #1: Computational efficiency**

This structure is not efficient on GPUs!

Why Transformer [1]?

Let's look at the issues with RNNs first!

Issue #1: Computational efficiency

This structure is not efficient on GPUs!

Solution #1: An architecture without sequential operations

Issue #2: Long-term dependencies
Issue #2: Long-term dependencies

sv: Viskleken | eng: Telephone (game)
Issue #2: Long-term dependencies

sv: Viskleken | eng: Telephone (game)

Our unpublished results:

Doesn't work if too many kids. (Sebastien, 189...)
Issue #2: Long-term dependencies

sv: Viskleken | eng: Telephone (game)

Our unpublished results:

*Doesn't work if too many kids.*

(Sebastien, 199...)

Solution #2: Always look at the whole sequence (or rather sub-sequence)
Transformer
Transformer: scaled dot-product self-attention

$T$ - the number of tokens/elements
$Q \in \mathbb{R}^{T \times d}$ - queries
$K \in \mathbb{R}^{T \times d}$ - keys
$V \in \mathbb{R}^{T \times d_v}$ - values
Transformer: scaled dot-product self-attention

\( T \) - the number of tokens/elements

\( Q \in \mathbb{R}^{T \times d} \) - queries

\( K \in \mathbb{R}^{T \times d} \) - keys

\( V \in \mathbb{R}^{T \times d_v} \) - values

**Self-attention** when all \( Q, K, \) and \( V \) are a function of the same input sequence \( X \).

- \( Q = f_q(X), K = f_k(X), V = f_v(X) \)
- Typically all linear functions
Transformer: scaled dot-product self-attention

- The number of tokens/elements: $T$
- Queries: $Q \in \mathbb{R}^{T \times d}$
- Keys: $K \in \mathbb{R}^{T \times d}$
- Values: $V \in \mathbb{R}^{T \times d_v}$

For each token (query!) compute the "compatibility" with other tokens (keys!):

$$\frac{QK^T}{\sqrt{d}} \rightarrow [\mathbb{R}^{T \times T}]$$

- Or any other $f(Q, K) \rightarrow \mathbb{R}^{T \times T}$
Transformer: scaled dot-product self-attention

- $T$ - the number of tokens/elements
- $Q \in \mathbb{R}^{T \times d}$ - queries
- $K \in \mathbb{R}^{T \times d}$ - keys
- $V \in \mathbb{R}^{T \times d_v}$ - values

Normalize the "compatibility" for each query

$$\text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right)$$
Transformer: scaled dot-product self-attention

- $T$ - the number of tokens/elements
- $Q \in \mathbb{R}^{T \times d}$ - queries
- $K \in \mathbb{R}^{T \times d}$ - keys
- $V \in \mathbb{R}^{T \times d_v}$ - values

Compute new values:

$$\text{attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d}} \right) V$$
Figure 1: The Transformer - model architecture.
Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. \( n \) is the sequence length, \( d \) is the representation dimension, \( k \) is the kernel size of convolutions and \( r \) the size of the neighborhood in restricted 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^2 \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_k(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>
Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. $n$ is the sequence length, $d$ is the representation dimension, $k$ is the kernel size of convolutions and $r$ the size of the neighborhood in restricted 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>$O(n^2 \cdot d)$</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>$O(\log_k(n))$</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>

track do the we ? keep of How order
Some details

Complexity

<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>$O(n^2 \cdot d)$</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>$O(\log_k(n))$</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>

Input encoded positions, e.g. based on the $\sin$ function

Learn input position embeddings

track do the we ? keep of How order
Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. \( n \) is the sequence length, \( d \) is the representation dimension, \( k \) is the kernel size of convolutions and \( r \) the size of the neighborhood in restricted 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^2 \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_k(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>

• Input encoded positions, e.g. based on the \( \sin \) function
• Learn input position embeddings

Multi-head attention

• More effective (controlling for \#params)
Some details

Complexity

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. \( n \) is the sequence length, \( d \) is the representation dimension, \( k \) is the kernel size of convolutions and \( r \) the size of the neighborhood in restricted 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^2 \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_k(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>

track do the we? keep of How order

- Input encoded positions, e.g. based on the \( \sin \) function
- Learn input position embeddings

Multi-head attention

- More effective (controlling for #params)

Interpretability?
### Results

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
<th>Training Cost (FLOPs)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EN-DE</td>
<td>EN-FR</td>
</tr>
<tr>
<td>ByteNet [18]</td>
<td>23.75</td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk [39]</td>
<td>24.6</td>
<td>39.2</td>
</tr>
<tr>
<td>GNMT + RL [38]</td>
<td>25.16</td>
<td>40.46</td>
</tr>
<tr>
<td>ConvS2S [9]</td>
<td>26.03</td>
<td>40.56</td>
</tr>
<tr>
<td>MoE [32]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep-Att + PosUnk Ensemble [39]</td>
<td>40.4</td>
<td></td>
</tr>
<tr>
<td>GNMT + RL Ensemble [38]</td>
<td>26.30</td>
<td>41.16</td>
</tr>
<tr>
<td>ConvS2S Ensemble [9]</td>
<td>26.36</td>
<td><strong>41.29</strong></td>
</tr>
<tr>
<td>Transformer (base model)</td>
<td>27.3</td>
<td>38.1</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td><strong>28.4</strong></td>
<td><strong>41.8</strong></td>
</tr>
</tbody>
</table>
Thank you!

Discussion!