Thinking Like Transformers

Gail Weiss, Yoav Goldberg, Eran Yahav

Supported by European Union’s Horizon 2020 research and innovation programme, grant agreement no. 802774 (iEXTRACT)
Transformers are very effective
Transformers are very effective
But we don't know how they work
Question
What is the computational model of the Transformer-encoder?
Can it solve “Reverse”?

abcde \(\mapsto\) edcba
Can it solve “Reverse”?

$abode \mapsto edcba$

"standard" programming language
Can it solve “Reverse”?

$abcde \rightarrow edcba$

"standard" programming language
Can it solve “Reverse”?

$abcde \rightarrow edcba$

"standard" programming language
Can it solve “Reverse”? 

abcde \rightarrow edcba
Can it solve “Reverse”?

\[ abcde \mapsto edcba \]

"standard" programming language
Can it solve “Reverse”?

Can it solve “Reverse”?

abcde $\mapsto$ edcba

"standard" programming language
Can it solve “Reverse”?  

\[
\begin{array}{ccccc}
\text{abcde} & \leftrightarrow & \text{edcba}
\end{array}
\]
Can it solve “Reverse”?

abcde $\mapsto$ edcba

"standard" programming language
Can it solve “Reverse”?

$abcde \mapsto edcba$

Outputs

"standard" programming language

$e \ d \ c \ b \ a$
Can it solve “Reverse”?  

abcde $\mapsto$ edcba

"standard" programming language
Can it solve “Reverse”?

\[
\begin{array}{c}
\text{abcde} \rightarrow \text{edcba}
\end{array}
\]

"standard" programming language

\[
\begin{array}{cccccc}
\text{e} & \text{d} & \text{c} & \text{b} & \text{a}
\end{array}
\]
RASP
Restricted Access Sequence Processing
RASP
Restricted Access Sequence Processing

Outputs

0 1 2 3 4

one by one

Input Embedding

Input

Positional Encoding

Multi-Head Attention

Add & Norm

Feed Forward

Add & Norm

N_x
RASP
Restricted Access Sequence Processing

Outputs

Add & Norm
Feed Forward

Add & Norm
Multi-Head Attention

Positional Encoding

Input Embedding

Inputs

one by one

0 1 2 3 4

0 1 2 3 4

2 2 2 2 2
RASP
Restricted Access Sequence Processing

Outputs

Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention

N

Positional Encoding
Input Embedding

Inputs

0 1 2 3 4
one by one

0 1 2 3 4

0 1 2 3 4
2 2 2 2 2

1 2 3 4 0
RASP
Restricted Access Sequence Processing

Symbolic: “A” instead of [0.1,-0.2,0.65,...???]
RASP
Restricted Access Sequence Processing

Primitive Sequences:

- tokens = [R,A,S,P]
- indices = [0,1,2,3]
- length = [4,4,4,4]
RASP
Restricted Access Sequence Processing

Primitive Sequences:

Elementwise Operations:

tokens = [R,A,S,P]
indices = [0,1,2,3]
length = [4,4,4,4]

indices*2 = [0,2,4,6]
indices+length = [4,5,6,7]
...

...
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],\Rightarrow) \quad \text{res} = \text{aggregate}(s, [4,6,8]) \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \quad 0 \Rightarrow 4 \Rightarrow [0,4,7] \]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],==) \]

\[
\begin{array}{cccc}
1 & 2 & 2 \\
0 & F & F & F \\
1 & T & F & F \\
2 & F & T & T \\
\end{array}
\]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1, 2, 2], [0, 1, 2], ==) \]

1 2 2
0 F F F
1 T F F
2 F T T
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[
s = \text{select}([1,2,2],[0,1,2],=)
\]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1, 2, 2], [0, 1, 2], \Rightarrow) \]

```
1 2 2
0 F F F
1 T F F
2 F T T
```
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],==) \]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],==) \]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],==) \]

1 2 2
0 F F F
1 T F F
2 F T T
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[
\begin{align*}
\textbf{s} &= \text{select}([1,2,2],[0,1,2],==) \\
\text{res} &= \text{aggregate}(\textbf{s}, [4,6,8])
\end{align*}
\]

\[
\begin{array}{ccc}
1 & 2 & 2 \\
0 & F & F & 4 & 6 & 8 & \Rightarrow & 0 \\
1 & T & F & F & T & F & F & 4 & 6 & 8 & \Rightarrow & 4 & \Rightarrow & [0,4,7] \\
2 & F & T & T & F & T & T & 4 & 6 & 8 & \Rightarrow & 7
\end{array}
\]
Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],=) \quad \text{res}=\text{aggregate}(s, [4,6,8]) \]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>4</th>
<th>6</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>7</td>
</tr>
</tbody>
</table>

\[ 1 \ 2 \ 2 \quad F \ F \ F \quad F \ F \ F \quad 4 \ 6 \ 8 \quad \Rightarrow \quad 0 \quad 4 \quad \Rightarrow \quad [0,4,7] \]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],==) \quad \text{res} = \text{aggregate}(s, [4,6,8]) \]

\[
\begin{array}{c|c|c|c}
\text{Outputs} & \text{Add & Norm} & \text{Feed Forward} & \text{Multi-Head Attention} \\
\hline
0 & F & F & F \\
1 & T & F & F \\
2 & F & T & T \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
1 & 2 & 2 & 4 & 6 & 8 \\
0 & F & F & F & F & F & F & F & 4 & 6 & 8 & 4 & 6 & 8 \Rightarrow & 0 \\
1 & T & F & F & T & F & F & 4 & 6 & 8 \Rightarrow & 4 & 7 \\
2 & F & T & T & F & T & T & 4 & 6 & 8 \Rightarrow & 7 & 0,4,7 \\
\end{array}
\]
RASP
Restricted Access Sequence Processing

Non-Elementwise Operations:

\[ s = \text{select}([1,2,2],[0,1,2],\Rightarrow) \]
\[ \text{res=aggregate}(s, [4,6,8]) \]

<table>
<thead>
<tr>
<th>s</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 2</td>
<td>F F F 4 6 8 =&gt; 0</td>
</tr>
<tr>
<td>0 F F F</td>
<td>T F F 4 6 8 =&gt; 4 =&gt; [0,4,7]</td>
</tr>
<tr>
<td>1 T F F</td>
<td>F T T 4 6 8 =&gt; 7</td>
</tr>
<tr>
<td>2 F T T</td>
<td>F T T 4 6 8 =&gt; 7</td>
</tr>
</tbody>
</table>

Diagram:
- Outputs
- Add & Norm
- Feed Forward
- Add & Norm
- Multi-Head Attention
- Positional Encoding
- Input Embedding
- Inputs
RASP
Restricted Access Sequence Processing

indices*2 = [0, 2, 4, 6]
indices+length = [4, 5, 6, 7]
RASP
Restricted Access Sequence Processing

indices*2 = [0,2,4,6]
indices+length = [4,5,6,7]

\( s = \text{select}([1,2,2],[0,1,2],==) \)
\( \text{res} = \text{aggregate}(s, [4,6,8]) \)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>0</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td>4</td>
<td>6</td>
</tr>
</tbody>
</table>
Can it solve “Reverse”? 
\( abcd \rightarrow edcba \)
Can it solve “Reverse”?

```
abcde \rightarrow edcba
```
Can it solve “Reverse”?

\[ \text{abcde} \rightarrow \text{edcba} \]
RASP
Restricted Access Sequence Processing

target = length-indices-1; = [3,2,1,0]
target = length-indices-1;
flip = select(target, indices, ==);

\[
\begin{align*}
\text{flip} &= \text{select}([3,2,1,0],[0,1,2,3],==) \\
&= \begin{pmatrix}
3 & 2 & 1 & 0 \\
0 & F & F & F & T \\
1 & F & F & T & F \\
2 & F & T & F & F \\
3 & T & F & F & F
\end{pmatrix}
\end{align*}
\]
**RASP**

Restricted Access Sequence Processing

target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);

reverse=aggregate(flip, [R,A,S,P])

<table>
<thead>
<tr>
<th></th>
<th>R</th>
<th>A</th>
<th>S</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>F F F T</td>
<td>R A S P</td>
<td>=&gt;</td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>F F T F</td>
<td>R A S P</td>
<td>=&gt;</td>
<td>S</td>
<td>=&gt;</td>
</tr>
<tr>
<td>F T F F</td>
<td>R A S P</td>
<td>=&gt;</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>T F F F</td>
<td>R A S P</td>
<td>=&gt;</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>
target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);
target = length−indices−1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);

RASP
Restricted Access Sequence Processing
target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);
target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);
target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);

RASP
Restricted Access Sequence Processing
RASP
Restricted Access Sequence Processing

\[
\text{target} = \text{length-indices-1};
\]
\[
\text{flip} = \text{select}(\text{target}, \text{indices}, ==);
\]
\[
\text{reverse} = \text{aggregate}(\text{flip}, \text{tokens});
\]
RASP
Restricted Access Sequence Processing

target = length-indices-1;
flip = select(target, indices, ==);
reverse = aggregate(flip, tokens);

Layer 1, Attention Head 1

Layer 2, Attention Head 1
Conclusion

- RASP: abstraction for transformer-encoder
- Solve formal tasks in transformer-encoders
  - Even Dyck-$k$ for arbitrary $k$ and unbounded depth!
- Translate to neural transformer-encoders architectures
  - Exact weights still required

More details in paper!

Try it out!! :

github.com/tech-srl/RASP