Do We Really Need That Many Parameters In Transformer For Extractive Summarization? Discourse Can Help!

Wen Xiao, Patrick Huber and Giuseppe Carenini
University of British Columbia
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM+20]

- Is it necessary to use heavy-weight dot-product self-attention in extractive summarization?

- Discourse trees are good indicators of importance in the text. [Mar99]

Applying discourse in the attention module might help reducing number of learnable parameters in the extractive summarization model.
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])
  - Can we reduce the number of parameter to train/finetune?

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM20]

- Discourse trees are good indicators of importance in the text. [Mar99]
  - Applying discourse in the attention module might help reducing number of learnable parameters in the extractive summarization model.
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])
  - Can we reduce the number of parameter to train/finetune?

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM+20]
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])
  - Can we reduce the number of parameter to train/finetune?

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM+20]
  - Is it necessary to use heavy-weight dot-product self-attention in extractive summarization?
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])
  - Can we reduce the number of parameter to train/finetune?

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM+20]
  - Is it necessary to use heavy-weight dot-product self-attention in extractive summarization?

- Discourse trees are good indicators of importance in the text. [Mar99]
Motivation

- The current summarization models are too large to train/finetune (e.g. BERTSUM: 118M [LL19])
  - Can we reduce the number of parameter to train/finetune?

- Light-weight attention modules have been proposed and applied on other tasks. [RST20][TBM^+20]
  - Is it necessary to use heavy-weight dot-product self-attention in extractive summarization?

- Discourse trees are good indicators of importance in the text. [Mar99]
  - Applying discourse in the attention module might help reducing number of learnable parameters in the extractive summarization model.
What is Extractive Summarization?

- Select units (e.g. EDUs, sentences, ...) that can best represent the whole document
- Can be regarded as a sequence labeling problem
Extractive Summarization Model

- **BERT Unit Encoder** to get unit representation (EDU/Sentence)
- **Synthesizer (generalized transformer)** based Document Encoder to encode the units
- **MLP Decoder** to get the importance score of each unit
Synthesizer - A Generalized Transformer

- Same structure as transformer
- It supports different attention modules [TBM\textsuperscript{+}20]
  (a) Dot-Product Self-Attention (original transformer)
  (b) Dense Self-Attention
  (c) Random Self-Attention (fixed or learnt)
  (d) No attention (baseline model)
- We propose another self-attention module: Discourse Tree Attention.

\[ A = \text{softmax}(Q(X) \cdot K(X)) \]  
\[ A = \text{softmax}(	ext{Dense}(X)) \]  
\[ A = \text{softmax}(	ext{Random}) \]  
\[ A = I \]
Discourse Tree Attention

- Fixed attention, as the embedding of discourse tree
- Three variants of tree-to-matrix encoding:
  - Dependency Tree (mainly nuclearity information)
  - Constituency Tree (structure information only)
  - Constituency Tree with Nuclearity (structure + nuclearity)
Most downstream applications for discourse use the transformed dependency trees over constituency trees [Mar99, HYN+13, XGCL20].

Step 1: constituency tree $\rightarrow$ dependency tree [HYN+13]

Step 2: dependency tree $\rightarrow$ attention matrix [XGCL20]
Encode the compositional structure of the document.

The closer the units are in the discourse tree, the more attention they should pay to each other.

$$M^L_{ij} = \begin{cases} 1, & \text{if } EDU_i \text{ and } EDU_j \text{ in the same constituent at level } L \\ 0, & \text{otherwise} \end{cases}$$
Encode the compositional structure of the document

The closer the units are in the discourse tree, the more attention they should pay to each other

\[ M^{L}_{ij} = \begin{cases} 
1, & \text{if } EDU_i \text{ and } EDU_j \text{ in the same constituent at level } L \\
0, & \text{otherwise}
\end{cases} \]
Variant #3 - Constituency Tree with Nuclearity

We also take Nuclearity into consideration

\[
M^{L}_{ij} = \begin{cases} 
2, & \text{if } EDU_i \text{ and } EDU_j \text{ in the same constituent at level } L \\
1, & \text{if } EDU_i \text{ and } EDU_j \text{ in the same constituent at level } L \\
0, & \text{otherwise}
\end{cases}
\]

Nucleus

Satellite

Normalized Attention Map with Nuclearity
Experiments - Settings

- Dataset: CNNDM

<table>
<thead>
<tr>
<th>#token/doc</th>
<th>#EDU/doc</th>
<th>#Sent/doc</th>
<th>#EDU(Oracle)</th>
<th>#Sent(Oracle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>546</td>
<td>70.2</td>
<td>27.2</td>
<td>6.4</td>
<td>3.1</td>
</tr>
</tbody>
</table>

- EDU Segmentor: top performing EDU Segmentor on RST-DT [WLY18]
- Discourse Parser: top performing Discourse Parser on RST-DT [WLW17]
- Evaluation Metric: ROUGE score
- We select the top 6 EDUs or top 3 sentences based on importance scores
- For all the models, we use two-layer synthesizer with 8 heads or single head.
- Hyper-parameter Setting can be found in the paper.
## Experiment Results - EDU Level

<table>
<thead>
<tr>
<th>Model</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
<th># Heads</th>
<th># Params (attn)</th>
<th># Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Setting ($d_k = d_v = d_q = 64, d_{inner} = 3072$)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dot Product (8)</td>
<td>41.02</td>
<td>18.78</td>
<td>37.96</td>
<td>8</td>
<td>3.2M</td>
<td>12.7M</td>
</tr>
<tr>
<td>Dot Product (1)</td>
<td>40.92‡</td>
<td>18.69‡</td>
<td>37.85‡</td>
<td>1</td>
<td>0.4M</td>
<td>9.9M</td>
</tr>
<tr>
<td>Dense</td>
<td>40.70</td>
<td>18.65†</td>
<td>37.74†</td>
<td>1</td>
<td>1.5M</td>
<td>11.0M</td>
</tr>
<tr>
<td>Learned Random</td>
<td>40.24</td>
<td>18.28</td>
<td>37.32</td>
<td>1</td>
<td>0.7M</td>
<td>10.3M</td>
</tr>
<tr>
<td>Fixed Random</td>
<td>40.36</td>
<td>18.35</td>
<td>37.40</td>
<td>1</td>
<td>0.2M</td>
<td>9.7M</td>
</tr>
<tr>
<td>No attention</td>
<td>39.89</td>
<td>17.98</td>
<td>36.99</td>
<td>1</td>
<td>0.2M</td>
<td>9.7M</td>
</tr>
<tr>
<td>D-Tree</td>
<td>40.43</td>
<td>18.32</td>
<td>37.45</td>
<td>1</td>
<td>0.2M</td>
<td>9.7M</td>
</tr>
<tr>
<td>C-Tree</td>
<td>40.80†</td>
<td>18.56</td>
<td>37.74†</td>
<td>1</td>
<td>0.2M</td>
<td>9.7M</td>
</tr>
<tr>
<td>C-Tree w/Nuc</td>
<td>40.76</td>
<td>18.59†</td>
<td>37.73</td>
<td>1</td>
<td>0.2M</td>
<td>9.7M</td>
</tr>
</tbody>
</table>
The C-Tree discourse tree attentions are better than all the other fixed attentions.
They are competitive with the single-head learned attentions with less learnable parameters in the attention mechanism.
The parameters in the attention module is only a small portion in the whole model, so we also test with a more balanced setting.
Experiment Results - EDU Level

<table>
<thead>
<tr>
<th>Model</th>
<th>Rouge-1</th>
<th>Rouge-2</th>
<th>Rouge-L</th>
<th># Heads</th>
<th># Params (attn)</th>
<th># Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balanced Models (d_k = d_v = d_q = 512, d_{inner} = 512)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dot Product(8)</td>
<td>40.95</td>
<td>18.52</td>
<td>37.78</td>
<td>8</td>
<td>25.2M</td>
<td>27M</td>
</tr>
<tr>
<td>Dot Product(1)</td>
<td>40.64</td>
<td>18.33</td>
<td>37.54</td>
<td>1</td>
<td>3.2M</td>
<td>4.8M</td>
</tr>
<tr>
<td>C-Tree w/Nuc</td>
<td>40.70</td>
<td>18.46†</td>
<td>37.63</td>
<td>1</td>
<td>1.6M</td>
<td>3.2M</td>
</tr>
</tbody>
</table>

In this setting, the C-Tree w/Nuc is better than the single-head dot-product attention, and is competitive with the 8-head dot-product attention.
C-tree discourse tree attentions achieves the best performance, and it’s significantly better than single-head/8-head Dot-Product attentions.
We extend and adapt the “Synthesizer” framework for extractive summarization by proposing a new discourse tree self-attention method.

The empirical results show that our fixed tree attentions are significantly better than other fixed attention baselines, and comparable with the learned attentions on both EDU level and Sentence level.
Future Work

▶ Explore ways to also incorporate rhetorical relations into discourse tree attention.

▶ The C-Tree with Nuclearity doesn’t perform better than C-Tree, which may suggest more exploration should be done in terms of the representation of nuclearity.

▶ Explore the combination of different kinds of learned and fixed attentions to see if it helps improving the performance.

▶ Instead of two-level encoder, inject the tree attentions directly to the BERT Document Encoder.
Thanks!


Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann, *Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation*. 

