Relational Attention Networks via Fully-Connected CRFs

Ziyin Liu‡, Junxiang Chen‡, Paul Pu Liang*, Masahito Ueda‡
‡University of Tokyo, *Northeastern University, **Carnegie Mellon University

Abstract
In this work, we propose a novel attention mechanism that offers a new level of interpretability.

Motivation
- Attention with structure and with mutual interactions - e.g. attention
- Interpretability
  - Allows us to visualize interactions
- Tractability and Practicality
  - End-to-End Trainable
  - Competitive computation speed

CRF Attention
- Attention can be defined as a weighted sum of the encoder hidden states:
  \[ e \leftarrow \sum a_i h_i \]
- In this work, we let define the attention as the states of a CRF with two states. The state 1 is interpreted to be “relevant” to the task; 0 means “irrelevant”:
  \[ a_i \in \{0, 1\} \]
- We make the CRF fully connected to allow the network to discover structure on its own (See figure 2). Previous methods use a predefined structure such a linear chain.

Mean Field Approximation
- Fully connected CRFs are not tractable
  - Finding the best state in \(2^d\) many states is slow
- Monte Carlo Procedures are not end-to-end trainable
  - We cannot backpropagate through a sampling procedure
- Mean field Approximation
  - End-to-end trainable; fast empirical convergence (within 10 iterations)

Algorithm 1 Mean-field-Attention(\(\omega, h_i, d_i\)). Note that \(D\) can be any arbitrarily defined function with necessary smoothness and regularity (e.g., a multilayer perceptron). \(\omega\) means concatenation.
1. Input: \((\omega, h_i, d_i)\)
2. Initialize \(t = 0\)
3. \(\phi_i \leftarrow D_{\text{mean}}(h_i, d_i)\) // computing on-site potential
4. \(J_{ij} \leftarrow D_{\text{pair}}(h_i, h_j, d_i)\) // computing interaction potential
5. \(\chi^{(0)} = \phi_i\)
6. while \(\chi^{(t)}\) not converged do
7. \(\langle \phi_i \rangle = \text{sigmoid}(\chi^{(t)} \phi_i)\)
8. \(\chi^{(t+1)} = \sum J_{ij} \langle \phi_j \rangle + \phi_i\)
9. \(t \leftarrow t + 1\)
10. Output: \(\text{sigmoid}(\chi^{(t-1)})\) // output expected spin at the converged point

Algorithm 2 Encoder-Decoder(\{\text{word}_i\})
1. Input: \{\text{word}_i\} // The input sequence such as a sequence of words.
2. Initialize \(d_0 = 0, t = 0\)
3. \(h_i \leftarrow \text{Encoder(\text{word}_i)}\) // For a sentence of length \(T\), \(i \in \{1, ..., T\}\)
4. while \(d_i \neq \text{END}\)
5. \(t \leftarrow t + 1\)
6. \(a_i^{(t)} \leftarrow \text{Mean-field-Attention}(\omega, h_i, d_i)\)
7. \(e \leftarrow \sum a_i h_i\)
8. \(d_i = \text{Decoder}(d_{i-1}, e)\)
9. Output: \{\text{word}_i\} // The sequence we want to output

EXPERIMENT RESULT
- Chunking Counting Task
  - Count the number of chunks of ones in the input
- Sorting task
  - Sort a sequence of integers of length 2 to 40

Table 1: Performance comparison (lower is better). For MFA, the subscript \(f\) refers to the maximum number of iterations allowed in the model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Counting</th>
<th>Sorting</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN [14]</td>
<td>720.32</td>
<td>-2.34</td>
</tr>
<tr>
<td>LSTM [12]</td>
<td>145.30</td>
<td>-0.13</td>
</tr>
<tr>
<td>Bidirectional LSTM [20]</td>
<td>215.91</td>
<td>-8.53</td>
</tr>
<tr>
<td>LSTM with Attention [3]</td>
<td>571.64</td>
<td>-18.43</td>
</tr>
<tr>
<td>VAE Attention</td>
<td>198.23</td>
<td>-8.44</td>
</tr>
<tr>
<td>MFA1</td>
<td>106.62</td>
<td>-20.76</td>
</tr>
<tr>
<td>MFA5</td>
<td>76.13</td>
<td>-25.12</td>
</tr>
</tbody>
</table>

Figure 3: (a) LSTM with Attention for Chunk Counting. Notice that the traditional attention mechanism outputs an attention that is highly imbalanced and hard to interpret. (b) MF-CRF Attention for Chunk Counting, notice that the attention correctly focuses on the chunk of 1 in a balanced way.

Figure 1: The y axis is negative log-likelihood loss, and the x is the length of testing sequence.

In this section, we show the interaction relationships between each of the inputs as a heat map, i.e. we plot \(J_{ij}\). This offers new interpretability for deep learning models. The following example shows \(J_{ij}\) for a sorting problem. Notice that the output has temporal dimension \(t \in 1, ..., T\) where \(T = 5\), so we expect to have 5 different \(J_{ij}\’s\) for each of the outputs. Notice that \(J_{ij}\) has a graphical structure and we plot a minimum spanning tree on \(J_{ij}\) to show the most significant interaction edges.

Figure 2: In this problem, the input is \{2, 9, 11, 17\}, and the network correctly sorts the input: \{1, 2, 9, 11, 17\}. At step 5, we see that all the other inputs have strongest interaction with 17, which is output of the last time step; this suggests that at this step, the network learned to not consider any interaction between the rest of input. It only needs to identify and output the largest element.