Ordered Neurons: Integrating Tree Structures Into Recurrent Neural Networks

Best paper at ICLR 2019

Mohammadali(Sobhan) Niknamian
CS886: Deep Learning and Natural Language Processing
Winter 2020
Motivation

• Underlying structure of language is usually tree-like

• Single words are composed to form meaningful larger units called “constituents”.

• Standard LSTM architecture does not have an explicit bias towards modeling a hierarchy of constituents.
How to predict the latent tree structure?

• Supervised Syntactic parser
  • This solution is limiting for several reasons:
    1) Few languages have annotated data for training such a parser.
    2) In some situations, syntactic rules tend to be broken (e.g. in tweets).
    3) Languages change over time, so syntax rules may evolve.

• Grammar induction: The task of learning the syntactic structure of language from raw corpora without access to expert-labeled data.
  • This is an open problem.
How to predict the latent tree structure?

• Recurrent Neural Networks (RNNs)
  • RNNs impose a chain structure on the data.
  • This assumption is in conflict with the latent non-sequential structure of language.
  • This gives rise to problems such as:
    • Capturing long-term dependencies
    • Achieving good generalization
    • Handling negation

• However, some evidence exist that traditional LSTMs with sufficient capacity may encode the tree structure implicitly.
How to predict the latent tree structure?

• Proposed method: ON-LSTM

  • Is able to differentiate the life cycle of information stored inside each of the neurons.
    • High ranking neurons will store long-term information which is kept for several steps.
    • Low ranking neurons will store short-term information that can be rapidly forgotten.

  • There is no strict division between high and low ranking neurons.
    • Neurons are actively allocated to store long/short information during each step of processing the input.
Requirements

• The hidden state $h_t$ of our model would ideally contain information from all nodes in the path between current input $x_t$ and the root $S$.
• Each node in the tree must be represented by a set of neurons in the hidden state.
• The model should dynamically reallocate the dimensions of the hidden state to each node.
Ordered neurons

• An inductive bias that forces neurons in the cell state of the LSTM to represent information at different time scales.
  • High ranking neurons contain long-term information
  • Low ranking neurons contain short-term information

• To erase (or update) high-ranking neurons, the model should first erase (or update) all lower-ranking neurons.

• The differentiation between low and high ranking neurons is learnt in a data-driven fashion and determined in each time step.
ON-LSTM: general architecture

- The new model uses an architecture similar to the standard LSTM.
- The only difference is in the update function of cell state $c_t$.

\[
\begin{align*}
    f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
    i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
    o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
    \hat{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
    h_t &= o_t \circ \tanh(c_t)
\end{align*}
\]

\[
\begin{align*}
    \tilde{f}_t &= \text{cumax}(W_{\tilde{f}} x_t + U_{\tilde{f}} h_{t-1} + b_{\tilde{f}}) \\
    \tilde{i}_t &= 1 - \text{cumax}(W_{\tilde{i}} x_t + U_{\tilde{i}} h_{t-1} + b_{\tilde{i}}) \\
    \omega_t &= \tilde{f}_t \circ \tilde{i}_t \\
    \hat{f}_t &= f_t \circ \omega_t + (\tilde{f}_t - \omega_t) = \tilde{f}_t \circ (f_t \circ \tilde{i}_t + 1 - \tilde{i}_t) \\
    \hat{i}_t &= i_t \circ \omega_t + (\tilde{i}_t - \omega_t) = \tilde{i}_t \circ (i_t \circ \tilde{f}_t + 1 - \tilde{f}_t) \\
    c_t &= \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{c}_t
\end{align*}
\]
Activation function: `cumax()`

- Input is a numerical vector
- It is the cumulative sum of the softmax of the input vector

\[ \hat{g} = \text{cumax}(\ldots) = \text{cumsum}(\text{softmax}(\ldots)) \]

- We could approximate this as a binary gate \( g = (0, \ldots, 0, 1, \ldots, 1) \).
- This binary gate splits the cell state into two segments: 0-segment and 1-segment.
- The model can apply different update rules on the two segment to differentiate long/short-term information.
Intuition behind new update rules

- We will explain this with an example:

\[ c_t = (\_, \_, \_, \_, \_, \_, \_, \_, \_) \]

\[ f_t = (0, 0, 0, 1, 1, 1, 1, 1, 1) \]

\[ \tilde{f}_t = (1, 1, 1, 1, 1, 0, 0, 0) \]

\[ \omega_t = (0, 0, 0, 1, 1, 1, 0, 0, 0) \]

\[ \hat{f}_t = (0, 0, 0, \_, \_, \_, 1, 1, 1) \]

\[ \hat{\tilde{f}}_t = \text{cmax}(W_{f} x_t + U_{f} h_{t-1} + b_{f}) \]

\[ \hat{i}_t = 1 - \text{cmax}(W_{i} x_t + U_{i} h_{t-1} + b_{i}) \]

\[ \omega_t = \hat{f}_t \circ \tilde{\tilde{i}}_t \]

\[ \hat{\tilde{f}}_t = f_t \circ \omega_t + (\hat{\tilde{f}}_t - \omega_t) = \hat{\tilde{f}}_t \circ (f_t \circ \tilde{\tilde{i}}_t + 1 - \tilde{\tilde{i}}_t) \]

\[ \hat{i}_t = \tilde{\tilde{i}}_t \circ \omega_t + (\tilde{\tilde{i}}_t - \omega_t) = \tilde{\tilde{i}}_t \circ (\tilde{\tilde{i}}_t \circ \hat{f}_t + 1 - \hat{f}_t) \]

\[ c_t = \hat{f}_t \circ c_{t-1} + \hat{i}_t \circ \hat{\tilde{c}}_t \]
Experiment: Language Modeling

- Perplexity on the Penn TreeBank (PTB) dataset.
- Perplexity measures the ability of a model in predicting the next word in a sentence (lower is better).

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>Validation</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zaremba et al. (2014) - LSTM (large)</td>
<td>66M</td>
<td>82.2</td>
<td>78.4</td>
</tr>
<tr>
<td>Gal &amp; Ghahramani (2016) - Variational LSTM (large, MC)</td>
<td>66M</td>
<td>-</td>
<td>73.4</td>
</tr>
<tr>
<td>Kim et al. (2016) - CharCNN</td>
<td>19M</td>
<td>-</td>
<td>78.9</td>
</tr>
<tr>
<td>Merity et al. (2016) - Pointer Sentinel-LSTM</td>
<td>21M</td>
<td>72.4</td>
<td>70.9</td>
</tr>
<tr>
<td>Grave et al. (2016) - LSTM</td>
<td>-</td>
<td>-</td>
<td>82.3</td>
</tr>
<tr>
<td>Grave et al. (2016) - LSTM + continuous cache pointer</td>
<td>-</td>
<td>-</td>
<td>72.1</td>
</tr>
<tr>
<td>Iinan et al. (2016) - Variational LSTM (tied) + augmented loss</td>
<td>51M</td>
<td>71.1</td>
<td>68.5</td>
</tr>
<tr>
<td>Zilly et al. (2016) - Variational RHN (tied)</td>
<td>23M</td>
<td>67.9</td>
<td>65.4</td>
</tr>
<tr>
<td>Zoph &amp; Le (2016) - NAS Cell (tied)</td>
<td>54M</td>
<td>-</td>
<td>62.4</td>
</tr>
<tr>
<td>Shen et al. (2017) - PRPN-LM</td>
<td>-</td>
<td>-</td>
<td>62.0</td>
</tr>
<tr>
<td>Melis et al. (2017) - 4-layer skip connection LSTM (tied)</td>
<td>24M</td>
<td>60.9</td>
<td>58.3</td>
</tr>
<tr>
<td>Merity et al. (2017) - AWD-LSTM - 3-layer LSTM (tied)</td>
<td>24M</td>
<td>60.0</td>
<td>57.3</td>
</tr>
<tr>
<td><strong>ON-LSTM - 3-layer (tied)</strong></td>
<td>25M</td>
<td>58.29 ± 0.10</td>
<td>56.17 ± 0.12</td>
</tr>
<tr>
<td><strong>Yang et al. (2017) - AWD-LSTM-MoS</strong></td>
<td>22M</td>
<td>56.5</td>
<td>54.4</td>
</tr>
</tbody>
</table>

PP(W) = $\sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1...w_{i-1})}}$
### Best models based on perplexity

<table>
<thead>
<tr>
<th>Model</th>
<th>Validation perplexity</th>
<th>Test perplexity</th>
<th>Number of params</th>
<th>Paper / Source</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mogrifier LSTM + dynamic eval (Mels et al., 2019)</td>
<td>44.9</td>
<td>44.8</td>
<td>24M</td>
<td>Mogrifier LSTM</td>
<td>Official</td>
</tr>
<tr>
<td>AdvSoft + AWD-LSTM-MoS + dynamic eval (Wang et al., 2019)</td>
<td>46.63</td>
<td>46.01</td>
<td>22M</td>
<td>Improving Neural Language Modeling via Adversarial Training</td>
<td>Official</td>
</tr>
<tr>
<td>FRAGE + AWD-LSTM-MoS + dynamic eval (Gong et al., 2018)</td>
<td>47.38</td>
<td>46.54</td>
<td>22M</td>
<td>FRAGE: Frequency-Agnostic Word Representation</td>
<td>Official</td>
</tr>
</tbody>
</table>

Experiment: Unsupervised Constituency Parsing

• Compares the latent tree structure induced by the model with those annotated by human experts.

• Lets consider $d_t^f$ be the split point in the master forget gate $\tilde{f}_t$ in time step $t$.

• We sort the $\{d_t^f\}$ in the decreasing order. For the first $d_t^f$ we split the sequence into constituents $((x_{<i}), (x_i, (x_{>i})))$. Then we repeat this recursively for constituents $(x_{<i})$ and $(x_{>i})$. 
Evaluating constituency parsing

Gold standard brackets: \textbf{S-(0:11), NP-(0:2), VP-(2:9), VP-(3:9), NP-(4:6), PP-(6-9), NP-(7,9), NP-(9:10)}

Candidate brackets: \textbf{S-(0:11), NP-(0:2), VP-(2:10), VP-(3:10), NP-(4:6), PP-(6-10), NP-(7,10)}

- Precision: $3/7 = 42.9\%$
- Recall: $3/8 = 37.5\%$
- F1: 40%
### Experiment: Unsupervised Constituency parsing

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Data</th>
<th>Training Object</th>
<th>Vocab Size</th>
<th>Parsing F1 $\mu$ (σ)</th>
<th>Depth WSJ</th>
<th>Accuracy on WSJ by Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRPN-UP</td>
<td>AllNLI Train</td>
<td>LM</td>
<td>76k</td>
<td>WSJ10 66.3 (0.8)</td>
<td>5.8</td>
<td>28.7</td>
</tr>
<tr>
<td>PRPN-LM</td>
<td>AllNLI Train</td>
<td>LM</td>
<td>76k</td>
<td>WSJ10 52.4 (4.9)</td>
<td>6.1</td>
<td>37.8</td>
</tr>
<tr>
<td>PRPN-UP</td>
<td>WSJ Train</td>
<td>LM</td>
<td>15.8k</td>
<td>WSJ10 62.2 (3.9)</td>
<td>5.8</td>
<td>24.8</td>
</tr>
<tr>
<td>PRPN-LM</td>
<td>WSJ Train</td>
<td>LM</td>
<td>10k</td>
<td>WSJ10 70.5 (0.4)</td>
<td>5.9</td>
<td>26.2</td>
</tr>
<tr>
<td>ON-LSTM 1st-layer</td>
<td>WSJ Train</td>
<td>LM</td>
<td>10k</td>
<td>WSJ10 35.2 (4.1)</td>
<td>5.6</td>
<td>38.1</td>
</tr>
<tr>
<td>ON-LSTM 2nd-layer</td>
<td>WSJ Train</td>
<td>LM</td>
<td>10k</td>
<td>WSJ10 65.1 (1.7)</td>
<td>5.6</td>
<td>46.2</td>
</tr>
<tr>
<td>ON-LSTM 3rd-layer</td>
<td>WSJ Train</td>
<td>LM</td>
<td>10k</td>
<td>WSJ10 54.0 (3.9)</td>
<td>5.3</td>
<td>44.8</td>
</tr>
<tr>
<td>300D ST-Gumbel w/o Leaf GRU</td>
<td>AllNLI Train</td>
<td>NLI</td>
<td>–</td>
<td>WSJ10 19.0 (1.0)</td>
<td>–</td>
<td>15.6</td>
</tr>
<tr>
<td>300D RL-SPINN w/o Leaf GRU</td>
<td>AllNLI Train</td>
<td>NLI</td>
<td>–</td>
<td>WSJ10 22.8 (1.6)</td>
<td>–</td>
<td>18.9</td>
</tr>
<tr>
<td>CCM</td>
<td>WSJ10 Full</td>
<td>–</td>
<td>–</td>
<td>71.9</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>DMV+CCM</td>
<td>WSJ10 Full</td>
<td>–</td>
<td>–</td>
<td>77.6</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>UML-DOP</td>
<td>WSJ10 Full</td>
<td>–</td>
<td>–</td>
<td>82.9</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Random Trees</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>31.7 (0.3)</td>
<td>5.3</td>
<td>17.4</td>
</tr>
<tr>
<td>Balanced Trees</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>43.4 (0.0)</td>
<td>4.6</td>
<td>22.1</td>
</tr>
<tr>
<td>Left Branching</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>19.6 (0.0)</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Right Branching</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>56.6 (0.0)</td>
<td>12.4</td>
<td>–</td>
</tr>
</tbody>
</table>
Experiment: Unsupervised Constituency parsing

Figure A.1: Left parses are from the 2nd layer of the ON-LSTM model, Right parses are converted from human expert annotations (removing all punctuations).
Experiment: Targeted Syntactic Evaluation

• A collection of tasks that evaluate language models along three different structure-sensitive linguistic phenomena:
  1) Subject-verb agreement
  2) Reflexive anaphora
  3) Negative polarity items

• Given a large number of minimally different pairs of a grammatical and an ungrammatical sentence, the model should assign higher probability to the grammatical sentence.

a. The bankers knew the officer smiles.
b. *The bankers knew the officer smile.

a. The bankers thought the pilot embarrassed himself.
b. *The bankers thought the pilot embarrassed themselves.

a. No authors that the security guards like have ever been famous.
b. *The authors that no security guards like have ever been famous.
Experiment: Targeted Syntactic Evaluation

- Long-term dependency means that an unrelated phrase exist between the targeted pairs of words.
- The paper states that the reason standard LSTM performs better on short-term dependencies is due to the small number units in the hidden states of the ON-LSTM, which is insufficient to take into account both long and short-term information.
References


Thank you!