**INTRODUCTION**

**Motivation**
- Williams et al. (2018): Gumbel-Softmax latent trees
- Outperform sequential RNNs.
- Generate non-syntactic and shallow trees.
- Generate unstable structures.
- This work
  - Includes experiments on 10 common NLP tasks.
  - Gives empirical explanations to above phenomena.
  - Helps understand the behavior of tree RNNs.

**BACKGROUND**

**TreeRNNs (Socher et al., 2011)**
- Parse tree LSTM (Zhu et al., 2015; Tai et al., 2015).
- Latent tree LSTM
  - by Gumbel-Softmax (Choi et al., 2018).
  - by reinforcement learning (Yogatama et al., 2017).
- N-ary balanced tree (Munkhdalai and Yu, 2017).

**Gumbel-Softmax (Jang et al., 2017)**
- Softmax: \( P(x_i | π) = \frac{\exp(σ(x_i))}{\sum_i \exp(σ(x_i))} \)
- Gumbel-Softmax Trick:
  \( g_i = x_i + (- \log(- \log(u_i))) \), \( u_i \sim U(0,1) \)
  \( P(\arg\max_i g_i = |g|) = P(x_i | π) \)
Can be applied to learn latent trees.

**CAN POOLING REPLACE TREES?**

**Pooling Mechanism**

**Experimental Result**

**MAIN RESULTS**

<table>
<thead>
<tr>
<th>Classification</th>
<th>Relation</th>
<th>Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>AGN</td>
<td>ARP</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>67.4</td>
<td>21.3</td>
</tr>
<tr>
<td>+max-pooling</td>
<td>71.8</td>
<td>21.6</td>
</tr>
<tr>
<td>+self-attention</td>
<td>72.5</td>
<td>21.2</td>
</tr>
<tr>
<td>Syntactic</td>
<td>68.6</td>
<td>22.2</td>
</tr>
<tr>
<td>+max-pooling</td>
<td>69.7</td>
<td>21.8</td>
</tr>
<tr>
<td>+self-attention</td>
<td>72.2</td>
<td>21.5</td>
</tr>
<tr>
<td>Balanced</td>
<td>69.6</td>
<td>22.3</td>
</tr>
<tr>
<td>+max-pooling</td>
<td>70.6</td>
<td>21.6</td>
</tr>
<tr>
<td>+self-attention</td>
<td>72.5</td>
<td>21.6</td>
</tr>
<tr>
<td>Left</td>
<td>67.7</td>
<td>21.6</td>
</tr>
<tr>
<td>+max-pooling</td>
<td>71.2</td>
<td>20.5</td>
</tr>
<tr>
<td>+self-attention</td>
<td>72.1</td>
<td>21.6</td>
</tr>
<tr>
<td>Right</td>
<td>68.7</td>
<td>23.1</td>
</tr>
<tr>
<td>+max-pooling</td>
<td>71.6</td>
<td>21.6</td>
</tr>
<tr>
<td>+self-attention</td>
<td>72.4</td>
<td>21.6</td>
</tr>
</tbody>
</table>

* All the tree structures above are associated with bidirectional leaf RNNs.

**WHY DO TRIVIAL TREES WORK BETTER?**

**Generation: Right Branching Tree Benefits from Structure**

**Classification: Binary Balanced Tree Benefits from Shallowness**

- \( ρ \)-Random Trees (\( ρ \in [0,1] \)):
  - \( Pr(n > \lfloor \frac{N}{2} \rfloor) = ρ \)
  - \( Pr(n < \lfloor \frac{N}{2} \rfloor) = 1 - ρ \)
- Randomly swap group sizes.
- Special Cases:
  - \( ρ = 1 \): balanced tree.
  - \( ρ = 0 \): Sequencial structure.

**CONCLUSIONS**

This work empirically shows
- Tree structure indeed helps in sentence modeling.
- However, syntax may not be the main contributor.
  - Trivial trees outperform binary constituency parse trees.
- LSTM units keep short-term information better.
- Tree models perform better with crucial words closer to the final representation.
- Structural advantages cannot be fully taken by pooling mechanisms.