Neural parsers no longer have much of the model structure provided to classical parsers. How do they perform so well without it?

**Why don’t we need a grammar?**

Adjacent tree labels are redundant with LSTM features.

If we can predict surrounding tree labels from our LSTM representation of the input, then this information doesn’t need to be provided explicitly by grammar production rules.

We find that for 92.3% of spans, the label of the span’s parent can be predicted from the neural representation of the span.

**Do we need tree constraints?**

Not for F1.

Many neural parsers no longer model output correlations with grammar rules, but still use output correlations from tree constraints.

Predicting span brackets independently gives nearly identical performance on PTB development set F1 and produces valid trees for 94.5% of sentences.

Almost a full point of F1 is lost by truncating context 5 words away from span endpoints and half a point with 10 words.

**Is distant context important?**

Yes!

Shuffling words outside a window around span endpoints hurts performance even with large context windows.

**What about lexicon features?**

The character LSTM captures the same information.

Heavily engineered lexicons used to be critical to good performance, but neural models typically don’t use them.

Word features from the Berkeley Parser (Petrov and Klein 2007) can be predicted with over 99.7% accuracy from the character LSTM representation.

**How do they perform so well without it?**

**What word representations do we need?**

A character LSTM is sufficient.

<table>
<thead>
<tr>
<th>Representation</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Only</td>
<td>91.44</td>
</tr>
<tr>
<td>Word and Tag</td>
<td>92.09</td>
</tr>
<tr>
<td>Character LSTM Only</td>
<td>92.24</td>
</tr>
<tr>
<td>Character LSTM and Word</td>
<td>92.22</td>
</tr>
<tr>
<td>Character LSTM, Word, and Tag</td>
<td>92.24</td>
</tr>
</tbody>
</table>

**Do LSTMs introduce useful inductive bias compared to feedforward networks?**

Yes!

We compare a truncated LSTM with feedforward architectures that are given the same inputs.

The LSTM outperformed the best feedforward by 6.5 F1.

Word Only | 91.44
Word and Tag | 92.09
Character LSTM Only | 92.24
Character LSTM and Word | 92.22
Character LSTM, Word, and Tag | 92.24