**Goals**

- Much research has been devoted to developing neural dependency parsers with complex, task-specific architectures
- Typical approach: use specialized neural networks to predict discrete actions in a dedicated, transition-based parsing algorithm

**SyntaxNet** AKA *Parsey McParseface* (Andor et al., 2016):

Feedforward network with beam search and CRF loss

Ablated RNN Grammar (Kuncoro et al., 2016): Stack-LSTM with bidirectional LSTM for phrase composition (SOTA)

- Can we get competitive (or even superior) parsing results with a simple architecture using general-purpose components?

**Dependency Parsing**

- Automatically annotate sentences, focusing on the functional role each phrase plays

  **Head**: Edge source, more contentful role (predicate → arguments)

  **Dependent**: Edge target

  **Label**: Edge type (Nominal Subject, Adjectival Clause)

- Particularly useful for NLU tasks, such as semantic parsing or knowledge base population
- Graph-based approach to parsing: assign weights to each possible edge, construct a maximum spanning tree

**LSTM**

Step one: BiLSTM over the sequence of word and part of speech tag embeddings, take all topmost LSTM states R (= \[ \text{stack}_i^m (r_i) \])

**Variable-class classification (= attention)**

- We want to predict heads (classes) given dependents (inputs), but the number of possible heads changes from sentence to sentence
- Thus, we want to predict \( P(y_{\text{edge}}) = j | r_i, r_j \)
- \( \text{softmax}(RU^{(1)}_i + RU^{(3)}_j) \) achieves this naturally

\[
P(y | r_i, r_j) \propto \exp (r_i^T U^{(1)} r_j) \exp (u^{(3)} r_j)
\]

- Relatively large network (other models use \(~100\) LSTM dims)
- Highly regularized with dropout
- Reducing Adam’s \( \beta_2 \) from .999 to .9 significantly improved performance (\( p < .05 \))

**Final model (edge scorer)**

- Everything is so big!
- We can get more control over the tradeoffs between speed, overfitting, and underfitting by shrinking \( r_i \) with smaller MLPs before the biaffine output layers (deep biaffine model as opposed to shallow biaffine)
- Result: four representations for each word
- Naturally reflects the intuition that the relationships we want to capture are asymmetric

\[
\text{Arc dep} \quad \text{Arc head} \quad \text{Label dep} \quad \text{Label head}
\]

**Hyperparameters**

<table>
<thead>
<tr>
<th>Param</th>
<th>Value</th>
<th>Param</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding size</td>
<td>100</td>
<td>Embedding dropout</td>
<td>33%</td>
</tr>
<tr>
<td>LSTM size</td>
<td>400</td>
<td>LSTM dropout</td>
<td>33%</td>
</tr>
<tr>
<td>Edge MLP size</td>
<td>500</td>
<td>Edge MLP dropout</td>
<td>33%</td>
</tr>
<tr>
<td>Label MLP size</td>
<td>100</td>
<td>Label MLP dropout</td>
<td>33%</td>
</tr>
<tr>
<td>Depth</td>
<td>3</td>
<td>MLP depth</td>
<td>1</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( 2e^{-3} )</td>
<td>( \beta_1, \beta_2 )</td>
<td>9</td>
</tr>
<tr>
<td>Annealing</td>
<td>.75 ( \text{max} )</td>
<td>( t_{\text{max}} )</td>
<td>50,000</td>
</tr>
</tbody>
</table>

**PTB Results**

- **Type**
  - SD 3.3.0
  - CTB

- **Model**
  - Ballesteros et al. (2016) 93.6 91.4 87.7 86.2
  - Transition Andor et al. (2016) 94.6 92.8 – –
  - Kuncoro et al. (2016) \textbf{95.8} \textbf{94.6} – –
  - Kiperwasser and Goldberg (2016) 93.9 91.9 87.6 86.1
  - Graph
  - Cheng et al. (2016) 94.1 91.5 88.1 85.7
  - Hashimoto et al. (2016) 94.7 92.9 – –
  - Deep biaffine 95.7 94.1 \textbf{89.3} 88.2

**CoNLL 09 Results**

- **Catalan**
  - UAS LAS UAS LAS UAS LAS
  - Andor et al. 92.7 89.8 84.7 80.9 88.9 84.6
  - Deep biaffine \textbf{94.7} \textbf{92.0} 88.9 85.4 92.1 \textbf{87.4}

- **Chinese**
  - UAS LAS UAS LAS UAS LAS
  - Andor et al. 93.2 91.2 90.9 89.2 92.6 90.0
  - Deep biaffine \textbf{95.2} \textbf{93.2} \textbf{93.5} 91.4 \textbf{94.3} 91.7

**Affect of classifier type (SD 3.5.0)**

- **Classifier**
  - UAS LAS Sents/sec
  - Deep biaffine \textbf{95.8} \textbf{94.2} 410.9
  - Shallow biaffine 95.7 94.0* 299.0
  - Shallow b. (50% MLP dropout) 95.7 94.1* 300.1
  - Shallow b. (300d LSTM) 95.6* 93.9* 373.2
  - Traditional attention 95.5* 93.9* 367.4

**Conclusion**

- Our simple, straightforward parser uses only neural components, effectively no task-specific architecture
- Substantially outperforms most more complex neural transition-based parsers
- Substantially outperforms all other neural graph-based parsers
- The biaffine approach to attention is theoretically justified, here beats the more traditional approach
- Adding final MLP layers to the LSTM helps to maximize speed and performance, captures head-dependent asymmetries
- This work provides a fast, simple, high-performing baseline against which to test more complex architectures

**Related work**

- Transition-based
  - Nivre et al. (2006): Feature-based
  - Chen and Manning (2014): First successful neural parser
  - Andor et al. (2014): Extend with beam search / CRF loss
  - Kuncoro et al. (2016): Extend with LSTMs (SOTA)

- Graph-based
  - McDonald and Pereira (2006): Feature-based
  - Kiperwasser and Goldberg (2016): First neural graph-based parser
  - Cheng et al. (2016): Keep track of previous decisions
  - Hashimoto et al. (2016): Jointly learn tagging & chunking

**References**


