A Minimal Span-Based Neural Constituency Parser

Mitchell Stern, Jacob Andreas, Dan Klein
UC Berkeley
 Parsing as Span Classification

She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
Minimality in Parsing

Grammar:
Minimality in Parsing

Grammar:

\[ S(\text{decided}) \rightarrow \text{NP(workers)} \text{ VP(decided)} \]

[Collins (1999)]
Minimality in Parsing

Grammar:

\[ S(\text{decided}) \rightarrow \text{NP}(\text{workers}) \quad \text{VP}(\text{decided}) \]  

\[ S \rightarrow \text{NP}^S \quad \text{VP}^S \]  

[Collins (1999)]  

[Klein and Manning (2003)]
Minimality in Parsing

Grammar:

\[
S(\text{decided}) \rightarrow \text{NP(} \text{workers} \text{)} \quad \text{VP(} \text{decided} \text{)}
\]

\[
S \quad \rightarrow \quad \text{NP}^S \quad \text{VP}^S
\]

\[
S \quad \rightarrow \quad \text{NP} \quad \text{VP}
\]

[Collins (1999)]
[Klein and Manning (2003)]
[Hall et al. (2014)]
Minimality in Parsing

Grammar:

\[ S(\text{decided}) \rightarrow NP(\text{workers}) \ VP(\text{decided}) \]
\[ S \rightarrow NP^S \ VP^S \]  \[\text{[Collins (1999)]}\]
\[ S \rightarrow NP \ VP \]  \[\text{[Klein and Manning (2003)]}\]
\[ S \]  \[\text{[Hall et al. (2014)]}\]
\[ S \]  \[\text{[Vinyals et al. (2015)]}\]
Minimality in Parsing

Scoring:
Minimality in Parsing

Scoring:

\[ \text{score}(S \rightarrow \text{NP}^S \text{ VP}^S) \]
Minimality in Parsing

Scoring:

\[
\text{score}(S \rightarrow \text{NP}^S \text{ VP}^S)
\]

\[
\text{score}(i, k, j, S \rightarrow \text{NP VP})
\]

[Klein and Manning (2003)]

[Hall et al. (2014)]
Minimality in Parsing

Scoring:

\[ \text{score}(S \rightarrow \text{NP}^S \text{ VP}^S) \]
\[ \text{score}(i, k, j, S \rightarrow \text{NP} \text{ VP}) \]
\[ \text{score}(i, j, S) \text{ and } \text{score}_{\text{action}}(i, k, j) \]
Minimality in Parsing

Scoring:

\[ \text{score}(S \rightarrow \text{NP}^S \text{ VP}^S) \]

\[ \text{score}(i, k, j, S \rightarrow \text{NP VP}) \]

\[ \text{score}(i, j, S) \text{ and } \text{score}_{\text{action}}(i, k, j) \]

\[ \text{score}(i, j, S) \]

[Klein and Manning (2003)]

[Hall et al. (2014)]

[Cross and Huang (2016)]

[This work]
Minimality in Parsing

Decoding:
Minimality in Parsing

Decoding:

Chart-based

Globally optimal, O(n³) time complexity
Minimality in Parsing

Decoding:

Chart-based
  Globally optimal, $O(n^3)$ time complexity

Transition-based
  Greedy, $O(n)$ or $O(n^2)$ time complexity
Tree Scoring Function

\[ s_{\text{tree}}(T) = \sum_{(\ell, (i, j)) \in T} s(i, j, \ell) \]
Tree Scoring Function

\[ s_{\text{tree}}(T) = \sum_{(\ell,(i,j)) \in T} s(i, j, \ell) \]

She enjoys playing tennis.
Tree Scoring Function

$$s_{\text{tree}}(T) = \sum_{(\ell, (i,j)) \in T} s(i, j, \ell) = s(0, 5, S)$$

She enjoys playing tennis.
Tree Scoring Function

She enjoys playing tennis.

\[
s_{\text{tree}}(T) = \sum_{(\ell, (i, j)) \in T} s(i, j, \ell)
\]

\[
= s(0, 5, \text{S}) + s(0, 1, \text{NP})
\]
Tree Scoring Function

\[ s_{\text{tree}}(T) = \sum_{(\ell, (i, j)) \in T} s(i, j, \ell) \]

\[ = s(0, 5, S) + s(0, 1, \text{NP}) + s(1, 4, \text{VP}) \]

She enjoys playing tennis.
Tree Scoring Function

\[ s_{\text{tree}}(T) = \sum_{(\ell, (i, j)) \in T} s(i, j, \ell) \]

\[ = s(0, 5, S) + s(0, 1, \text{NP}) + s(1, 4, \text{VP}) + s(2, 4, S-\text{VP}) \]

She enjoys playing tennis.
Tree Scoring Function

She enjoys playing tennis.

$$s_{\text{tree}}(T) = \sum_{(\ell,(i,j)) \in T} s(i, j, \ell)$$

$$= s(0, 5, S) + s(0, 1, NP) + s(1, 4, VP) + s(2, 4, S-VP) + s(3, 4, NP)$$
$s_{\text{best}}(i, i + 1) = \max_{\ell} [s(i, i + 1, \ell)]$
Dynamic Program: Base Case

\[ s_{\text{best}}(i, i + 1) = \max_{\ell} [s(i, i + 1, \ell)] \]

Pick best label
$$s_{\text{best}}(i, j) = \max_{\ell} [s(i, j, \ell)]$$

$$+ \max_{k} [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)]$$
Dynamic Program: General Case

Pick best label

\[ s_{\text{best}}(i, j) = \max_{\ell} [s(i, j, \ell)] + \max_k [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)] \]
Dynamic Program: General Case

\[ s_{\text{best}}(i, j) = \max_{\ell} [s(i, j, \ell)] \]

\[ + \max_{k} [s_{\text{best}}(i, k) + s_{\text{best}}(k, j)] \]

Pick best split point
Scoring Function Implementation

[Inspired by Cross and Huang (2016)]
Scoring Function Implementation

She enjoys playing tennis.

[Inspired by Cross and Huang (2016)]
She enjoys playing tennis.

[Inspired by Cross and Huang (2016)]
Scoring Function Implementation

\[(f_j - f_i, b_i - b_j)\]

\[(f_i, b_i) \quad (f_j, b_j)\]

[Inspired by Cross and Huang (2016)]
Scoring Function Implementation

\[ s \rightarrow s(i, j, X) \]

\[ (f_j - f_i, b_i - b_j) \]

\[ (f_i, b_i) \quad (f_j, b_j) \]

Feedforward Network

Span Difference

Bidirectional LSTM

[Inspired by Cross and Huang (2016)]
Training

Want \( s_{\text{tree}}(T^*) > s_{\text{tree}}(T) \) for all \( T \neq T^* \)
Training

Want \( s_{\text{tree}}(T^*) > s_{\text{tree}}(T) \) for all \( T \neq T^* \)

Require larger margin for higher loss:

\[
 s_{\text{tree}}(T^*) \geq \Delta(T, T^*) + s_{\text{tree}}(T)
\]
Training

Want $s_{\text{tree}}(T^*) > s_{\text{tree}}(T)$ for all $T \neq T^*$

Require larger margin for higher loss:

$$s_{\text{tree}}(T^*) \geq \Delta(T, T^*) + s_{\text{tree}}(T)$$

Use hinge penalty function:

$$\max \left( 0, \Delta(\hat{T}, T^*) - s_{\text{tree}}(T^*) + s_{\text{tree}}(\hat{T}) \right)$$
Training

Use loss-augmented decoding during training:

\[ \hat{T} = \max_T \left[ \Delta(T, T^*) + s_{\text{tree}}(T) \right] \]
Use loss-augmented decoding during training:

\[
\hat{T} = \max_T [\Delta(T, T^*) + s_{\text{tree}}(T)]
\]

Loss-augmented decoding for Hamming loss:

Replace \( s(i, j, \ell) \) with \( s(i, j, \ell) + 1(\ell \neq \ell_{ij}^*) \)
## Initial Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall et al. (2014)</td>
<td>89.2</td>
</tr>
<tr>
<td>Vinyals et al. (2015)</td>
<td>88.3</td>
</tr>
<tr>
<td>Cross and Huang (2016)</td>
<td>91.3</td>
</tr>
<tr>
<td>Dyer et al. (2016)</td>
<td>91.7</td>
</tr>
<tr>
<td>Liu and Zhang (2017)</td>
<td>91.7</td>
</tr>
<tr>
<td>Our Chart Parser</td>
<td>91.7</td>
</tr>
</tbody>
</table>
Top-Down Parsing

She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
She enjoys playing tennis.
Top-Down Parsing

She enjoys playing tennis.
Top-Down Parsing

She enjoys playing tennis

S

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>She</td>
<td>enjoys</td>
</tr>
</tbody>
</table>

VP

<table>
<thead>
<tr>
<th>NP</th>
<th>VP</th>
</tr>
</thead>
<tbody>
<tr>
<td>playing</td>
<td>NP</td>
</tr>
<tr>
<td>tennis</td>
<td>She</td>
</tr>
</tbody>
</table>
She enjoys playing tennis.
She enjoys playing tennis.

Top-Down Parsing
Top-Down Parsing

She enjoys playing tennis.
She enjoys playing tennis.
Top-Down Parsing

She enjoys playing tennis.
She enjoys playing tennis.
Top-Down Parsing

She enjoys playing tennis.
\hat{\ell} = \operatorname{arg\,\max}_{\ell} \, s_{\text{label}}(i, j, \ell)
\[ \hat{\ell} = \arg\max_{\ell} [s_{\text{label}}(i, j, \ell)] \]

\[ \hat{k} = \arg\max_{k} [s_{\text{span}}(i, k) + s_{\text{span}}(k, j)] \]
Top-Down Training

Margin constraint for each decision:

score(gold) ≥ 1 + score(other)
Margin constraint for each decision:

\[ \text{score(gold)} \geq 1 + \text{score(other)} \]

Train with exploration using a dynamic oracle

[Goldberg and Nivre (2012), Cross and Huang (2016)]
# Initial Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall et al. (2014)</td>
<td>89.2</td>
</tr>
<tr>
<td>Vinyals et al. (2015)</td>
<td>88.3</td>
</tr>
<tr>
<td>Cross and Huang (2016)</td>
<td>91.3</td>
</tr>
<tr>
<td>Dyer et al. (2016)</td>
<td>91.7</td>
</tr>
<tr>
<td>Liu and Zhang (2016)</td>
<td>91.7</td>
</tr>
<tr>
<td>Our Chart Parser</td>
<td>91.7</td>
</tr>
<tr>
<td>Our Top-Down Parser</td>
<td>91.6</td>
</tr>
</tbody>
</table>
Extensions

Label scoring for unary chains:
• Split unary chains into top-middle-bottom
Extensions

Label scoring for unary chains:
• Split unary chains into top-middle-bottom

Structured label loss for unary chains:
• Hamming distance on labels (vs. 0-1 loss)
Extensions

Label scoring for unary chains:
• Split unary chains into top-middle-bottom

Structured label loss for unary chains:
• Hamming distance on labels (vs. 0-1 loss)

Split-based (vs. span-based) scoring:
• Left-right, concatenate, deep biaffine

[Cross and Huang (2016)] [Dozat and Manning (2016)]
## Final Results

<table>
<thead>
<tr>
<th>Parser</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hall et al. (2014)</td>
<td>89.2</td>
</tr>
<tr>
<td>Vinyals et al. (2015)</td>
<td>88.3</td>
</tr>
<tr>
<td>Cross and Huang (2016)</td>
<td>91.3</td>
</tr>
<tr>
<td>Dyer et al. (2016)</td>
<td>91.7</td>
</tr>
<tr>
<td>Liu and Zhang (2016)</td>
<td>91.7</td>
</tr>
<tr>
<td>Our Best Chart Parser</td>
<td>91.8</td>
</tr>
<tr>
<td>Our Best Top-Down Parser</td>
<td>91.8</td>
</tr>
</tbody>
</table>
Conclusion
Conclusion

A minimal span-based parser can achieve state-of-the-art results.
Conclusion

A minimal span-based parser can achieve state-of-the-art results.

Little is lost going from global to greedy decoding.
Conclusion

A minimal span-based parser can achieve state-of-the-art results.

Little is lost going from global to greedy decoding.

Various extensions yield only minimal gains beyond the core system.
Thanks!