Span-Based Constituency Parsing with Provably Optimal Dynamic Oracles

James Cross and Liang Huang
Oregon State University

EMNLP, Austin, TX
November 2, 2016
**Dependency vs. Constituency**

<table>
<thead>
<tr>
<th></th>
<th>search</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang &amp; Nivre 2011</td>
<td>beam</td>
<td>92.9</td>
</tr>
<tr>
<td>Chen &amp; Manning 2014</td>
<td>greedy</td>
<td>91.8</td>
</tr>
<tr>
<td>Zhou et al. (2015)</td>
<td>beam</td>
<td>93.3</td>
</tr>
<tr>
<td>Weiss et al. (2015)</td>
<td>beam</td>
<td>94.0</td>
</tr>
<tr>
<td>our work (ACL 2016)</td>
<td>greedy</td>
<td>93.4</td>
</tr>
<tr>
<td>Andor et al. (2016)</td>
<td>beam</td>
<td>94.4</td>
</tr>
</tbody>
</table>

**Red = Neural**

<table>
<thead>
<tr>
<th></th>
<th>search</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carreras et al. (2008)</td>
<td>cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Shindo et al. (2012)</td>
<td>cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Thang et al. (2015) (A*)</td>
<td>~cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Watanabe et al. (2015)</td>
<td>beam</td>
<td>90.7</td>
</tr>
<tr>
<td>Vinyals et al. (2015) (WSJ)</td>
<td>beam</td>
<td>90.5</td>
</tr>
</tbody>
</table>
Dependency vs. Constituency

<table>
<thead>
<tr>
<th>Model</th>
<th>Search</th>
<th>UAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang &amp; Nivre 2011</td>
<td>beam</td>
<td>92.9</td>
</tr>
<tr>
<td>Chen &amp; Manning 2014</td>
<td>greedy</td>
<td>91.8</td>
</tr>
<tr>
<td>Zhou et al. (2015)</td>
<td>beam</td>
<td>93.3</td>
</tr>
<tr>
<td>Weiss et al. (2015)</td>
<td>beam</td>
<td>94.0</td>
</tr>
<tr>
<td>our work (ACL 2016)</td>
<td>greedy</td>
<td>93.4</td>
</tr>
<tr>
<td>Andor et al. (2016)</td>
<td>beam</td>
<td>94.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Search</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carreras et al. (2008)</td>
<td>cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Shindo et al. (2012)</td>
<td>cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Thang et al. (2015) (A*)</td>
<td>~cubic</td>
<td>91.1</td>
</tr>
<tr>
<td>Watanabe et al. (2015)</td>
<td>beam</td>
<td>90.7</td>
</tr>
<tr>
<td>Vinyals et al. (2015) (WSJ)</td>
<td>beam</td>
<td>90.5</td>
</tr>
<tr>
<td>This Work</td>
<td>greedy</td>
<td>91.3</td>
</tr>
</tbody>
</table>

Red = Neural
Outline

- Span-Based Constituency Parsing
- Bi-Directional LSTM Span Features
- Provably Optimal Dynamic Oracle
- Experiments
Span-Based Parsing

- Previous work uses tree structures on stack
- We simplify to operate directly on sentence spans
- Simple-to-implement linear-time parsing

Previous work:

Stack

NP

I/PRP do/MD like/VBP

VP'

previous work

Queue

eating/VBG fish/NN

Our work:

Stack

I/PRP do/MD like/VBP

0 1 3 4 5

Queue

eating/VBG fish/NN
<table>
<thead>
<tr>
<th>Structural (even step)</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Combine</td>
</tr>
<tr>
<td>Label (odd step)</td>
<td>Label-X</td>
</tr>
<tr>
<td></td>
<td>No-Label</td>
</tr>
</tbody>
</table>

Current brackets $t = \{\}$

```
I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
0 1 2 3 4 5
```
<table>
<thead>
<tr>
<th>Structural (even step)</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Combine</td>
</tr>
<tr>
<td><strong>Label (odd step)</strong></td>
<td>Label-X</td>
</tr>
<tr>
<td></td>
<td>No-Label</td>
</tr>
</tbody>
</table>

The diagram shows a sentence structure with the following nodes:

- **S** (Sentence)
  - **NP** (Noun Phrase)
    - **PRP** (Personal Pronoun)
    - **MD** (Modal Verb)
    - **VBP** (Base Form Verb)
  - **VP** (Verb Phrase)
    - **VBG** (Base Form Verb)
    - **NN** (Noun)

The current brackets `t = {}` indicate the initial state of the parsing process.

The process includes:

- **Shift**
- **Combine**
- **Label-X**
- **No-Label**

The blue arrows represent the movement of tokens during the parsing process, starting from the root `S` and recursively breaking down into `NP` and `VP` components.
Structural (even step)
Shift
Combine

Label (odd step)
Label-X
No-Label

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
Shift

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

Label-NP

current brackets  t = {}

0 1 2 3 4 5

0 1 2 3 4 5

Structural (even step)
Shift
Combine

Label (odd step)
Label-X
No-Label

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
Shift

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

Label-NP

current brackets  t = {}
Structural (even step)  
Shift  
Combine

Label (odd step)  
Label-X  
No-Label

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

0  1  2  3  4  5

Shift

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

0  1  2  3  4  5

Shift

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN

0  1  2  3  4  5

current brackets  t = {}

Label-NP  t = {0NP1}
Structural (even step)  
Shift  
Combine  
Label-X  
No-Label  

Label (odd step)  

\[
\text{Structural (even step)} \quad \text{Shift} \quad \text{Combine} \quad \text{Label-X} \quad \text{No-Label}
\]

\[
\text{Label (odd step)}
\]

\[
\begin{align*}
&I/\text{PRP} \quad \text{do/MD} \quad \text{like/\text{VBP}} \quad \text{eating/\text{VBG}} \quad \text{fish/\text{NN}} \\
&\text{current brackets} \quad t = \{\}
\end{align*}
\]

\[
\begin{align*}
&I/\text{PRP} \quad \text{do/MD} \quad \text{like/\text{VBP}} \quad \text{eating/\text{VBG}} \quad \text{fish/\text{NN}} \\
&t = \{0\text{NP}_1\}
\end{align*}
\]

\[
\begin{align*}
&I/\text{PRP} \quad \text{do/MD} \quad \text{like/\text{VBP}} \quad \text{eating/\text{VBG}} \quad \text{fish/\text{NN}} \\
&t = \{0\text{NP}_1\}
\end{align*}
\]
Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

Current brackets: \( t = {} \)

1. Shift
2. Label-NP
3. No-Label
4. Shift
5. Shift
Structural (even step)
Shift
Combine
Label (odd step)
Label-X
No-Label

Current brackets
\( t = \{ \} \)

Shift
\( t = \{0\text{NP}_1 \} \)
Label-NP
\( t = \{0\text{NP}_1 \} \)
No-Label
\( t = \{0\text{NP}_1 \} \)
No-Label
"I/PRP
d/MD
like/VBP
eating/VBG
fish/NN
0
1
2
3
4
5
S
VP
NP
MD
do
VBP
I
like
eating
NN
fish

Shift
Shift
Shift
Shift
No-Label
No-Label
<table>
<thead>
<tr>
<th>Structural (even step)</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Combine</td>
</tr>
<tr>
<td>Label (odd step)</td>
<td>Label-X</td>
</tr>
<tr>
<td></td>
<td>No-Label</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Token</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I/PRP</td>
</tr>
<tr>
<td>1</td>
<td>do/MD</td>
</tr>
<tr>
<td>2</td>
<td>like/VBP</td>
</tr>
<tr>
<td>3</td>
<td>eating/VBG</td>
</tr>
<tr>
<td>4</td>
<td>fish/NN</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

$$t = \{0\text{NP}_1\}$$

**Diagram:**

```
S
  | VP
  | NP
  |   MD
  |   VBP
  |     VBG
  |     NP
  |     eating
  |     NN
  |     fish
```

- **NP**
- **VP**
- **S**
- **MD**
- **VBP**
- **VBG**
- **NN**
- **fish**
Structural (even step)

<table>
<thead>
<tr>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>Combine</td>
</tr>
<tr>
<td>Label-X</td>
</tr>
<tr>
<td>No-Label</td>
</tr>
</tbody>
</table>

Label (odd step)

<table>
<thead>
<tr>
<th>I/PRP</th>
<th>do/MD</th>
<th>like/VBP</th>
<th>eating/VBG</th>
<th>fish/NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

\[
t = \{0NP_1\}
\]
Structural (even step)
Shift
Combine
Label (odd step)
Label-X
No-Label

I/PRP do/MD like/VBP eating/VBG fish/NN
t = \{0NP_1\}

I/PRP do/MD like/VBP eating/VBG fish/NN
t = \{0NP_1\}
Structural (even step)  Shift
Combine

Label (odd step)  Label-X
No-Label

```
I/PRP  do/MD  like/VBP
0      1      2

eating/VBG  fish/NN
3      4      5

t = \{NP\}
```

Combine

```
I/PRP  do/MD  like/VBP
0      1      2

eating/VBG  fish/NN
3      4      5

t = \{NP\}
```

No-Label

```
I/PRP  do/MD  like/VBP
0      1      2

eating/VBG  fish/NN
3      4      5

t = \{NP\}
```
Structural (even step)

Shift

Combine

Label (odd step)

Label-X

No-Label

I/PRP  do/MD  like/VBP

0 1 2

eating/VBG  fish/NN

3 4 5

t = \{0NP_1\}

Combine

I/PRP  do/MD  like/VBP

0 1 2

eating/VBG  fish/NN

3 4 5

No-Label  t = \{0NP_1\}

Shift

I/PRP  do/MD  like/VBP

0 1 2

eating/VBG  fish/NN

3 4 5

No-Label  t = \{0NP_1\}
Structural (even step)     Shift

Combine

Label (odd step)     Label-X

No-Label

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRP</td>
<td>do/MD</td>
<td>like/VBP</td>
<td>eating/VBG</td>
<td>fish/NN</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ t = \{0NP_1\} \]

Combine

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRP</td>
<td>do/MD</td>
<td>like/VBP</td>
<td>eating/VBG</td>
<td>fish/NN</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ t = \{0NP_1\} \]

No-Label

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRP</td>
<td>do/MD</td>
<td>like/VBP</td>
<td>eating/VBG</td>
<td>fish/NN</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ t = \{0NP_1\} \]

Shift

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRP</td>
<td>do/MD</td>
<td>like/VBP</td>
<td>eating/VBG</td>
<td>fish/NN</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ t = \{0NP_1\} \]

Shift

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRP</td>
<td>do/MD</td>
<td>like/VBP</td>
<td>eating/VBG</td>
<td>fish/NN</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Structural (even step) | Shift
---|---
Combine
Label (odd step) | Label-X
No-Label

I/PRP  do/MD  like/VBP |
0 1 2

eating/VBG  fish/NN |
3 4 5
t = \{0NP_1\}

I/PRP  do/MD  like/VBP |
0 1

eating/VBG  fish/NN |
3 4 5

No-Label  t = \{0NP_1\}

Shift

I/PRP  do/MD  like/VBP |
0 1

eating/VBG  fish/NN |
3 4 5

No-Label  t = \{0NP_1\}

Shift

I/PRP  do/MD  like/VBP |
0 1

eating/VBG  fish/NN |
3 4 5

No-Label  t = \{0NP_1\}

Shift

I/PRP  do/MD  like/VBP |
0 1

eating/VBG  fish/NN |
3 4 5

Label-NP  t = \{0NP_1, 4NP_5\}
Structural (even step)  Shift
Combine
Label (odd step)  Label-X
No-Label

I/PRP  do/MD  like/VBP  eating/VBG  fish/NN
0  1  3  4  5

t = {0NP, 4NP}

S
NP  MD  VBP
PRP  do  like

VP
S
NP  VBG  fish
NN
Structural (even step) | Shift
---|---
| Combine
Label (odd step) | Label-
---|---
| No-Label

\[ t = \{0_{NP}, 4_{NP}\} \]
Structural (even step)  
Shift  
Combine
Label (odd step)  
Label-X  
No-Label

I/PRP  
do/MD  like/VBP  
eating/VBG  
fish/NN  
0  1  3  4  5

Combine

I/PRP  
do/MD  like/VBP  
eating/VBG  
fish/NN  
0  1  3  4  5

Label-S-VP  
t = \{0NP_1, 4NP_5, 3S_5, 3VP_5\}

S  
NP  
PRP  I

VP  
MD  VBP  I
do  like

S  
VP  
VBG  NP  
eating  NN  
fish

t = \{0NP_1, 4NP_5\}
Structural (even step)

Shift

Combine

Label (odd step)

Label-X

No-Label

I/PRP  do/MD like/VBP eating/VBG fish/NN

0 1 3 4 5

Combine

I/PRP  do/MD like/VBP eating/VBG fish/NN

0 1 3 4 5

Combine

I/PRP  do/MD like/VBP eating/VBG fish/NN

0 1 3 4 5
Structural (even step)

- Shift
- Combine

Label (odd step)

- Label-X
- No-Label

- $t = \{0NP_1, 4NP_5\}$

- $t = \{0NP_1, 4NP_5, 3S_5, 3VP_5\}$

- $t = \{0NP_1, 4NP_5, 3S_5, 3VP_5, 1VP_5\}$
Structural (even step)
- Shift
- Combine

Label (odd step)
- Label-X
- No-Label

Diagram:
- S
  - NP
  - VP
    - PRP
    - VBP
    - I
    - do
    - like
    - VBG
    - eating
    - NP
    - NN
    - fish

Trees:
- t = \{NP_1, NP_5\}
- t = \{NP_1, NP_5, S_5, VP_5\}
- t = \{NP_1, NP_5, S_5, VP_5, VP_5\}
Structural (even step)
Shift
Combine
Label (odd step)
Label-X
No-Label

$\text{I/PRP do/MĐ like/VBP eating/VBG fish/NN}$

$\text{t} = \{0\text{NP}_{1}, 4\text{NP}_{5}\}$

$\text{Label-S-VP}$
$t = \{0\text{NP}_{1}, 4\text{NP}_{5}, 3\text{S}_{5}, 3\text{VP}_{5}\}$

$\text{Combine}$

$\text{I/PRP do/MĐ like/VBP eating/VBG fish/NN}$

$\text{t} = \{0\text{NP}_{1}, 4\text{NP}_{5}, 3\text{S}_{5}, 3\text{VP}_{5}, 1\text{VP}_{5}\}$

$\text{Label-VP}$
$t = \{0\text{NP}_{1}, 4\text{NP}_{5}, 3\text{S}_{5}, 3\text{VP}_{5}, 1\text{VP}_{5}, 0\text{S}_{5}\}$

$\text{Combine}$

$\text{I/PRP do/MĐ like/VBP eating/VBG fish/NN}$

$\text{t} = \{0\text{NP}_{1}, 4\text{NP}_{5}, 3\text{S}_{5}, 3\text{VP}_{5}, 1\text{VP}_{5}, 0\text{S}_{5}\}$

$\text{Label-S}$
Advantages of Span-Based System

• Linear-time and fixed number of steps (well-suited for beam search)

• Separates prediction of structure and labels

• Predicts rules of arbitrary arity with no binarization
Advantages of Span-Based System

• Linear-time and fixed number of steps (well-suited for beam search)

• Separates prediction of structure and labels

• Predicts rules of arbitrary arity with no binarization

Q: How to decide which action to take? What features represent spans?
• Span-Based Constituency Parsing

• **Bi-Directional LSTM Span Features**
  • Provably Optimal Dynamic Oracle
  • Experiments
Sentence segment “eating fish” represented by two vectors:
- Forward component: $f_5 - f_3$ (Wang and Chang, ACL 2016)
- Backward component: $b_3 - b_5$
Span Features for Structure Action

to predict:

`Combine`

4 bi-LSTM span features

(no tree-structure information used)
Span Features for Label Action

to predict: Label-VP

3 bi-LSTM span features
(no tree-structure information used)
Training Scheme: Local

- Every parser state is paired with a correct action
- Separate multilayer perceptron for each action type
- Baseline training scheme (static oracle) uses canonical order with short-stack preference
Training Scheme: Local

- Every parser state is paired with a correct action
- Separate multilayer perceptron for each action type
- Baseline training scheme (static oracle) uses canonical order with short-stack preference

Correct action after mistake?
Outline

- Span-Based Constituency Parsing
- Bi-Directional LSTM Span Features
- **Provably Optimal Dynamic Oracle**
- Experiments
Dynamic Oracle: Motivation

• Static oracle training assumes all correct actions

• What to do after decoding mistakes?

gold path
Dynamic Oracle: Motivation

- Static oracle training assumes all correct actions
- What to do after decoding mistakes?
- Need a way to decide best action in arbitrary state: **Dynamic Oracle** (everywhere-defined optimal policy)

![](gold_path_diagram)
do I like eating fish.

Dynamic Oracle: Example
I like eating fish.
I do like eating fish.

Dynamic Oracle: Example.

smallest reachable gold bracket incl. $s_0$. 

The diagram shows a parse tree with labeled nodes representing parts of speech (e.g., NP, VP, PRP, MD, VBP, VBG, NP, NN) and words (e.g., I, do, like, eating, fish). The highlighted part of the tree indicates the smallest reachable gold bracket including $s_0$. 
do I like eating fish

Dynamic Oracle: Example

smallest reachable gold bracket incl. $s_0$
Dynamic Oracle: Example

I do like eating fish

smallest reachable gold bracket incl. $s_0$

next reachable

S

NP

PRP

VP

MD

VBP

S

VP

VBG

NP

NN

S

VP

VBG

NP

NN

I do like eating fish

S_1 S_0
I like eating fish.
I do like eating fish.
do I like eating fish?
do I like eating fish

Dynamic Oracle: Example

smallest reachable gold bracket incl. $s_0$

next reachable
Dynamic Oracle: Example

Dynamic Oracle: Combine
Dynamic Oracle: Example

I do like eating fish.
Dynamic Oracle: Example
Dynamic Oracle: Example

I do like eating fish.

smallest reachable
gold bracket incl. $s_0$

next reachable

I do like eating fish.
Dynamic Oracle: Example

I do like eating fish

Dynamic Oracle: Shift

smallest reachable gold bracket incl. $s_0$

next reachable

$s_0$
• Structure actions depend on next **reachable** bracket in gold tree
• All non-bracket label states —> **No-Label**
• All gold-bracket label states —> Correct label(s)
Dynamic Oracle: Full Definition

- Structure actions depend on next reachable bracket in gold tree.
- All non-bracket label states $\rightarrow$ No-Label.
- All gold-bracket label states $\rightarrow$ Correct label(s).

Gold Brackets

Current Brackets
Dynamic Oracle: Full Definition

- Structure actions depend on next reachable bracket in gold tree.
- All non-bracket label states $\rightarrow$ No-Label.
- All gold-bracket label states $\rightarrow$ Correct label(s).
Dynamic Oracle: Full Definition

- Structure actions depend on next reachable bracket in gold tree
- All non-bracket label states $\rightarrow$ No-Label
- All gold-bracket label states $\rightarrow$ Correct label(s)
Dynamic Oracle: Optimality/Complexity

• First provably optimal oracle for constituency parsing (optimal in both precision and recall)

• After each action next reachable may (or may not) be updated by tracing parent link in gold tree

• Also $O(n)$ steps, thus amortized $O(1)$ time

• Dependency parsing oracle (arc-std): worst case $O(n^3)$ per step
Dynamic Oracle: Optimality/Complexity

- First provably optimal oracle for constituency parsing (optimal in both precision and recall)
- After each action next reachable may (or may not) be updated by tracing parent link in gold tree
- Also $O(n)$ steps, thus amortized $O(1)$ time
- Dependency parsing oracle (arc-std): worst case $O(n^3)$ per step
Dynamic Oracle: Optimality/Complexity

• First provably optimal oracle for constituency parsing (optimal in both precision and recall)

• After each action next reachable may (or may not) be updated by tracing parent link in gold tree

• Also $O(n)$ steps, thus amortized $O(1)$ time

• Dependency parsing oracle (arc-std): worst case $O(n^3)$ per step
Training with Dynamic Oracle

<table>
<thead>
<tr>
<th>(scores on PTB 22)</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
</tbody>
</table>
Training with Dynamic Oracle

• Basic dynamic oracle: follow current model

<table>
<thead>
<tr>
<th>(scores on PTB 22)</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
</tbody>
</table>
Training with Dynamic Oracle

- Basic dynamic oracle: follow current model

<table>
<thead>
<tr>
<th>(scores on PTB 22)</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
<tr>
<td>Dynamic Oracle</td>
<td>91.14</td>
<td>91.61</td>
<td>91.38</td>
</tr>
</tbody>
</table>
Training with Dynamic Oracle

- Basic dynamic oracle: follow current model
- Problem: overfits training data, making fewer mistakes than test

<table>
<thead>
<tr>
<th>(scores on PTB 22)</th>
<th>Recall</th>
<th>Prec.</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
<tr>
<td>Dynamic Oracle</td>
<td>91.14</td>
<td>91.61</td>
<td>91.38</td>
</tr>
</tbody>
</table>
Training with Dynamic Oracle

• Basic dynamic oracle: follow current model

• Problem: overfits training data, making fewer mistakes than test

• Exploration: sample from softmax distribution (Ballesteros et al., 2016) to encourage more mistakes

<table>
<thead>
<tr>
<th>(scores on PTB 22)</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
<tr>
<td>Dynamic Oracle</td>
<td>91.14</td>
<td>91.61</td>
<td>91.38</td>
</tr>
</tbody>
</table>
Training with Dynamic Oracle

- Basic dynamic oracle: follow current model
- Problem: overfits training data, making fewer mistakes than test
- Exploration: sample from softmax distribution (Ballesteros et al., 2016) to encourage more mistakes

<table>
<thead>
<tr>
<th></th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Oracle</td>
<td>91.34</td>
<td>91.43</td>
<td>91.38</td>
</tr>
<tr>
<td>Dynamic Oracle</td>
<td>91.14</td>
<td>91.61</td>
<td>91.38</td>
</tr>
<tr>
<td>Dynamic + Exploration</td>
<td>91.07</td>
<td>92.22</td>
<td>91.64</td>
</tr>
</tbody>
</table>
Outline

• Span-Based Constituency Parsing
• Bi-Directional LSTM Span Features
• Provably Optimal Dynamic Oracle
• Experiments
• 50-dim word and 20-dim tag embeddings
• No pre-training
• Each LSTM layer 200 units each direction
• 200 ReLU units for each of structure and label predictors
## Results on Penn Treebank

<table>
<thead>
<tr>
<th>Parser</th>
<th>Search</th>
<th>Recall</th>
<th>Prec.</th>
<th>F&lt;sub&gt;1&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carreras et al. (2008)</td>
<td>cubic</td>
<td>90.7</td>
<td>91.4</td>
<td>91.1</td>
</tr>
<tr>
<td>Shindo et al. (2012)</td>
<td>cubic</td>
<td></td>
<td></td>
<td>91.1</td>
</tr>
<tr>
<td>Thang et al. (2015)</td>
<td>~cubic</td>
<td></td>
<td></td>
<td>91.1</td>
</tr>
<tr>
<td>Watanabe et al. (2015)</td>
<td>beam</td>
<td></td>
<td></td>
<td>90.7</td>
</tr>
<tr>
<td><strong>Static Oracle</strong></td>
<td>greedy</td>
<td>90.7</td>
<td>91.4</td>
<td>91.0</td>
</tr>
<tr>
<td><strong>Dynamic + Exploration</strong></td>
<td>greedy</td>
<td>90.5</td>
<td>92.1</td>
<td><strong>91.3</strong></td>
</tr>
</tbody>
</table>

- State of the art despite: simple system with greedy actions and small embeddings trained from scratch
Parsing Morphologically Rich Languages

```
SENT
  NP-ATS  VN  NP-SUJ  PONCT
    DETWH  V  DET  NC  
      Quelles  sont  les  perspectives
```

- **lemma = perspective**
- **coarse_POS = N**
- **gender = feminine**
- **number = plural**
- **subcategory = common**
Results on French Treebank

- Morphological feature embeddings (10 dim. each)
- Additional input to recurrent network
- For French, we used SPMRL 2014 predicted features

<table>
<thead>
<tr>
<th>Parser</th>
<th>Recall</th>
<th>Prec.</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Björkelund et al. (2014)</td>
<td></td>
<td></td>
<td>82.53</td>
</tr>
<tr>
<td>Static Oracle</td>
<td>83.50</td>
<td>82.87</td>
<td>83.18</td>
</tr>
<tr>
<td>Dynamic + Exploration</td>
<td>81.90</td>
<td>84.77</td>
<td>83.31</td>
</tr>
</tbody>
</table>
Summary

• Simple, easy-to-implement span-based parsing system

• No tree/label information in features (good candidate for dynamic programming)

• Linear time parsing with greedy decoding

• No pre-trained embeddings, small architecture, and minimal hyper-parameter tuning (trained on CPU)

• First optimal dynamic oracle for constituency parsing