Non-Deterministic Oracles for Unrestricted Non-Projective Transition-Based Dependency Parsing

Anders Björkelund and Joakim Nivre

anders@ims.uni-stuttgart.de

July 27, 2015
Table of Contents

Introduction

Oracles

Experiments

Conclusion
Motivation

- Recent progress in greedy transition-based dependency parsing using dynamic oracles
  - Statistical model trained to select the next best transition, after making a local mistake
Motivation

- Recent progress in **greedy** transition-based dependency parsing using *dynamic oracles*
  - Statistical model trained to select the *next best transition*, after making a local mistake

- **Search-based** transition-based parsers (beam search/DP) – trained to find optimal *sequence of transitions*
Motivation

- Recent progress in **greedy** transition-based dependency parsing using *dynamic oracles*
  - Statistical model trained to select the *next best transition*, after making a local mistake

- **Search-based** transition-based parsers (beam search/DP) – trained to find optimal *sequence of transitions*
  - Globally trained model, dynamic oracles not entirely applicable
Motivation

- Recent progress in **greedy** transition-based dependency parsing using *dynamic oracles*
  - Statistical model trained to select the *next best transition*, after making a local mistake

- **Search-based** transition-based parsers (beam search/DP) – trained to find optimal *sequence of transitions*
  - Globally trained model, dynamic oracles not entirely applicable

- Can spurious ambiguity be exploited to increase accuracy of search-based parsers?
Arc Standard system

- **Stack** of partially processed tokens

![Stack diagram with tokens s₀, s₁, s₂, ...]
Arc Standard system

- **Stack** of partially processed tokens
- **Buffer** of remaining input tokens

![Diagram showing Stack and Buffer]
Arc Standard system

- **Stack** of partially processed tokens
- **Buffer** of remaining input tokens
- Transitions:
  - **Shift** (SH)

![Diagram of Arc Standard system]

Buffer

\[ b_0 \quad b_1 \quad b_2 \quad ... \]

Stack

\[ s_0 \quad s_1 \quad s_2 \quad ... \]
Arc Standard system

- **Stack** of partially processed tokens
- **Buffer** of remaining input tokens
- Transitions:
  - Shift (SH)
  - LeftArc (LA)

![Diagram of Arc Standard system]

- **Stack**
  - $s_0$
  - $s_1$
  - $s_2$
  - ...

- **Buffer**
  - $b_0$
  - $b_1$
  - $b_2$
  - ...


Arc Standard system

- **Stack** of partially processed tokens
- **Buffer** of remaining input tokens
- Transitions:
  - Shift (SH)
  - LeftArc (LA)
  - RightArc (RA)

Buffer:

\[
\begin{array}{ccc}
  b_0 & b_1 & b_2 \\
\end{array}
\]

Stack:

\[
\begin{array}{c}
  s_0 \\
  s_1 \\
  s_2 \\
  \vdots
\end{array}
\]
Initial and Terminal states

- **Initial state** – root on stack, input on buffer

![Diagram](image-url)

- **Terminal state** – only root on stack, empty buffer
Initial and Terminal states

- **Initial state** – root on stack, input on buffer
- **Terminal state** - only root on stack, empty buffer
Example

root  John  likes  Mary
Example parse

\[
\text{root} \quad \text{John} \enspace \text{likes} \enspace \text{Mary}
\]

Buffer

\[
\text{John} \enspace \text{likes} \enspace \text{Mary}
\]

Stack

\[
\text{root}
\]

History:
Example parse

\[\text{root} \ \text{John \ likes \ Mary}\]

Buffer

\[\text{likes \ Mary}\]

Stack

\[\text{root} \ \text{John}\]

History: SH
Example parse

root  John  likes  Mary

Buffer

Mary

Stack

likes  John

root

History: SH SH
Example parse

root John likes Mary

Stack

likes root

Buffer

Mary

History: SH SH LA
Example parse

root John likes Mary

Buffer

Stack

Mary likes root

History: SH SH LA SH
Example parse

\[ root \text{ John likes Mary} \]

Buffer

Stack

likes \text{ root}

History: SH SH LA SH RA
Example parse

root  John  likes  Mary

Buffer

Stack

root

History: SH SH LA SH RA RA
Example parse

\textit{root} John likes Mary

\begin{center}
\begin{tikzpicture}
  \node [draw] {Buffer};
  \node [below] at (current bounding box.center) {John likes Mary};
  \node [left] at (current bounding box.center) {Stack};
  \node [draw] at (current bounding box.center) {root};
  \node [below right] at (current bounding box.center) {History: SH SH LA SH RA RA};
\end{tikzpicture}
\end{center}
Example parse

*root* John likes Mary

Stack

John

*root*

Buffer

likes Mary

History: SH SH LA SH RA RA

History: SH
Example parse

\[ \text{root} \text{ John likes Mary} \]

Stack

\[ \text{likes} \text{ John root} \]

Buffer

Mary

History: SH SH LA SH RA RA

History: SH SH
Example parse

\[ \text{root} \ John \ likes \ Mary \]

Stack

\[
\begin{array}{l}
Mary \\
likes \\
John \\
root
\end{array}
\]

Buffer

History: SH SH LA SH RA RA

History: SH SH SH
Example parse

$\text{root} \quad \text{John} \quad \text{likes} \quad \text{Mary}$

Buffer

Stack

likes
John
$\text{root}$

History: SH SH LA SH RA RA

History: SH SH SH RA
Example parse

\[ \text{root} \quad \text{John} \quad \text{likes} \quad \text{Mary} \]

Buffer

Stack

likes \text{root}

History: SH SH LA SH RA RA

History: SH SH SH RA LA
Example parse

Stack

Buffer

History:  SH SH LA SH RA RA

History:  SH SH SH RA LA RA

root  John  likes  Mary

root
Ambiguity as a lattice

The possible transition sequences can be illustrated as a lattice

The SH-LA ambiguity a *spurious ambiguity*
Dealing with non-projectivity

- Non-projective trees cannot be drawn without crossing edges

- Treatment: introduce new transition swap (SW) that moves the second stack item back onto the buffer (Nivre, 2009)

- Increases the amount of spurious ambiguity considerably
Lattice for non-projective sentence

root Ausgelöst wurde sie durch Intel

Corresponding lattice
Static oracle

1: if $\text{CANLA}(c, x)$ then
2: \hspace{1em} return LA
3: else if $\text{CANRA}(c, x)$ then
4: \hspace{1em} return RA
5: else
6: \hspace{1em} return SH
1: if $\text{CANLA}(c, x)$ then
2:     return LA
3: else if $\text{CANRA}(c, x)$ then
4:     return RA
5: else
6:     return SH
1: if CANLA(c, x) then
2:   return LA
3: else if CANRA(c, x) then
4:   return RA
5: else
6:   return SH

▶ Spurious ambiguity resolved by order of if-clauses
1: \textbf{if} \ CANLA(c, x) \textbf{then} \\
2: \hspace{1em} \textbf{return} \ LA \\
3: \textbf{else if} \ CANRA(c, x) \textbf{then} \\
4: \hspace{1em} \textbf{return} \ RA \\
5: \textbf{else if} \ CANSW(c, x) \textbf{then} \\
6: \hspace{1em} \textbf{return} \ SW \\
7: \textbf{else} \\
8: \hspace{1em} \textbf{return} \ SH
CanSwap

- Relies on the notion of projective order, obtained by in-order traversal

\[ root_0 \rightarrow \text{Ausgelöst}_1 \rightarrow \text{wurde}_2 \rightarrow \text{sie}_3 \rightarrow \text{durch}_4 \rightarrow \text{Intel}_5 \]
CanSwap

- Relies on the notion of projective order, obtained by in-order traversal

```
root0 Ausgelöst1 wurde2 sie3 durch4 Intel5
```

```
root0 Ausgelöst1 durch4 Intel5 wurde2 sie3
```
CanSwap

- Relies on the notion of projective order, obtained by in-order traversal

```
root_0  Ausgelöst_1  wurde_2  sie_3  durch_4  Intel_5
```

```
root_0  Ausgelöst_1  durch_4  Intel_5  wurde_2  sie_3
```
CanSwap

- Nivre (2009) swap as soon as possible (EAGER)
  $\Rightarrow$ leads to many unnecessary swaps

- Nivre et al. (2009) block some swaps when more substructure can be built (LAZY)
  $\Rightarrow$ still not always minimal
Potential spurious ambiguities

- Possible
  - SH-LA
  - SH-RA
  - SH-SW
- Impossible
  - LA-RA – *(implies cycle)*
  - SW-RA – *(violates projective order)*
  - SW-LA – *(violates projective order)*
  - And any superset of these
CanShift?

- Static oracles define when LA, RA, SW are permissible
- SH treated as fallback

- Simple solution:
  try and see if the correct parse can be recovered using EAGER
Can now build complete lattices

- With tests for all transitions we can construct lattices
- Cover *all* possible spurious ambiguities
- Searching the lattice for the shortest path yields minimally swapping oracle (\text{TMINIMAL})
Non-deterministic oracles

- Allow all possible spurious ambiguities (Nd-All)
- Allow only SH-Sw ambiguity (Nd-Sw)
Table of Contents

Introduction

Oracles

Experiments

Conclusion
Oracles

- Static
  - EAGER – (Nivre, 2009)
  - LAZY – (Nivre et al., 2009)
  - MINIMAL – new

- Non-deterministic
  - Nd-All – new
  - Nd-Sw – new
Data and Evaluation

Data

- SPRML Shared Task: Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish
- English: Penn Treebank converted to Stanford dependencies
- Standard splits train/dev/test

Evaluation

- Labeled Attachment Score (LAS)
- Significance Testing: Wilcoxon signed rank test

\[ \text{Significance Testing: } W < 0.05, \quad W^* < 0.01 \]
Data and Evaluation

Data

- SPRML Shared Task: Arabic, Basque, French, German, Hebrew, Hungarian, Korean, Polish, Swedish
- English: Penn Treebank converted to Stanford dependencies
- Standard splits train/dev/test

Evaluation

- Labeled Attachment Score (LAS)
- Significance Testing: Wilcoxon signed rank test

\[ \dagger < 0.05, \ \ddagger < 0.01 \]
### Data set stats (training data)

<table>
<thead>
<tr>
<th>% proj.</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
</tr>
</tbody>
</table>
Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
</tr>
</tbody>
</table>

Wide range of projectivity: German (alot) to Korean (none)
# Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>LAZY red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
</tr>
</tbody>
</table>

Reduction of swaps from **EAGER** to **LAZY**
Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>LAZY red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
</tr>
</tbody>
</table>

Reduction of swaps from **EAGER** to **LAZY**

- Reduces swaps by up to 80% (Arabic), 75% for German
- Corroborates results by Nivre et al. (2009)
Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>LAZY red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
</tr>
</tbody>
</table>

**Reduction of swaps from **EAGER** to **LAZY**

- Reduces swaps by up to 80% (Arabic), 75% for German
- Corroborates results by Nivre et al. (2009)
- Extremely few non-proj arcs in French and Hebrew since they are basically projective
## Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>Lazy red.</th>
<th>Minimal red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td>-</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td>-</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td>-</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td>-</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td>-</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
</tr>
</tbody>
</table>

Reduction of swaps from **Eager** to **Minimal**
## Data set stats (training data)

<table>
<thead>
<tr>
<th>Language</th>
<th>% proj.</th>
<th>Lazy red.</th>
<th>Minimal red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td>-</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td>-</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td>-</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td>-</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td>-</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
</tr>
</tbody>
</table>

Reduction of swaps from **EAGER** to **MINIMAL**

- **Lazy** already minimal in several cases
## Data set stats (training data)

<table>
<thead>
<tr>
<th>Language</th>
<th>% proj.</th>
<th>Lazy red.</th>
<th>Minimal red.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td>-</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td>-</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td>-</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td>-</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td>-</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
</tr>
</tbody>
</table>

Reduction of swaps from EAGER to MINIMAL

- **Lazy** already minimal in several cases
- Reduction relative to **Lazy** very small
## Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>Lazy red.</th>
<th>Minimal red.</th>
<th>unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
<td>9.94</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
<td>7.81</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td>-</td>
<td>1.31</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td>-</td>
<td>1.06</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td>-</td>
<td>2.66</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td>-</td>
<td>2.82</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
<td>10.25</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
<td>0.27</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td>-</td>
<td>10.57</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
<td>7.28</td>
</tr>
</tbody>
</table>

Amount of sentences without spurious ambiguity
**Data set stats (training data)**

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>Lazy red.</th>
<th>Minimal red.</th>
<th>unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
<td>9.94</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
<td>7.81</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td>-</td>
<td>1.31</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td>-</td>
<td>1.06</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td>-</td>
<td>2.66</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td>-</td>
<td>2.82</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
<td>10.25</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td>-</td>
<td>0.27</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td>-</td>
<td>10.57</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
<td>7.28</td>
</tr>
</tbody>
</table>

Amount of sentences without spurious ambiguity

- Only 10% without spurious ambiguity
## Data set stats (training data)

<table>
<thead>
<tr>
<th></th>
<th>% proj.</th>
<th>LAZY red.</th>
<th>MINIMAL red.</th>
<th>unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>97.32</td>
<td>80.59</td>
<td>80.79</td>
<td>9.94</td>
</tr>
<tr>
<td>de</td>
<td>67.23</td>
<td>75.09</td>
<td>83.88</td>
<td>7.81</td>
</tr>
<tr>
<td>en</td>
<td>99.90</td>
<td>71.92</td>
<td></td>
<td>1.31</td>
</tr>
<tr>
<td>eu</td>
<td>94.71</td>
<td>53.46</td>
<td></td>
<td>1.06</td>
</tr>
<tr>
<td>fr</td>
<td>99.97</td>
<td>16.67</td>
<td></td>
<td>2.66</td>
</tr>
<tr>
<td>he</td>
<td>99.82</td>
<td>8.33</td>
<td></td>
<td>2.82</td>
</tr>
<tr>
<td>hu</td>
<td>87.75</td>
<td>51.07</td>
<td>54.24</td>
<td>10.25</td>
</tr>
<tr>
<td>ko</td>
<td>100.00</td>
<td>-</td>
<td></td>
<td>0.27</td>
</tr>
<tr>
<td>pl</td>
<td>99.54</td>
<td>59.34</td>
<td></td>
<td>10.57</td>
</tr>
<tr>
<td>sv</td>
<td>93.62</td>
<td>75.90</td>
<td>77.79</td>
<td>7.28</td>
</tr>
</tbody>
</table>

Amount of sentences without spurious ambiguity

- Only 10% without spurious ambiguity
- Despite being projective, Korean still lots of ambiguity
Training (static)

- Greedy parser
  - Averaged perceptron (Collins, 2002)

- Beam search parser
  - Passive-aggressive algorithm (Crammer et al., 2006)
  - Using max-violation updates (Huang et al., 2012)
  - Averaging (Collins, 2002)
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Greedy –
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Greedy – next transition $t$ latent
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Greedy – next transition $t$ latent

Given current weights $w$, and state $c$

Latent gold

$$\tilde{t} = \arg \max_{t \in \text{Nd-Oracle}(c)} \text{score}(t, w)$$
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Greedy – next transition $t$ latent

Given current weights $w$, and state $c$

Latent gold

\[ \hat{t} = \arg \max_{t \in \text{Nd-Oracle}(c)} \text{score}(t, w) \]

Prediction

\[ \hat{t} = \arg \max_{t \in \text{Permissible}(c)} \text{score}(t, w) \]
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide
- Beam search
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Beam search – transition sequence $z$ latent
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Beam search – transition sequence $z$ latent

Given current weights $w$, and sentence $x$

Latent Gold

$$\tilde{z} = \arg\max_{z \in \text{Nd-Oracle}(x)} \text{score}(z, w)$$
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Beam search – transition sequence $z$ latent

Given current weights $w$, and sentence $x$

Latent Gold

$$\tilde{z} = \arg\max_{z \in \text{Nd-Oracle}(x)} \text{score}(z, w)$$

Prediction

$$\hat{z} = \arg\max_{z \in \text{Possible}(x)} \text{score}(z, w)$$
Training (non-deterministic)

- What is the “correct” solution to update against?
- Leave it latent – let the current parameters decide

- Beam search – transition sequence $z$ latent

Given current weights $w$, and sentence $x$

Latent Gold

$$
\hat{z} = \arg \max_{z \in \text{Nd-Oracle}(x)} \text{score}(z, w)
$$

Prediction

$$
\hat{\tilde{z}} = \arg \max_{z \in \text{Possible}(x)} \text{score}(z, w)
$$

Approximate search with beam search (beam size 20)
Problem 1: Most oracles generally extremely close

Problem 2: Performance on dev set not monotonically increasing as a function of training iterations

Solution: Tune number of iterations on dev data for each oracle

Final evaluation (test set): best static oracle vs best non-deterministic oracle
Problem 1: Most oracles generally extremely close
Problem 2: Performance on dev set not monotonically increasing as a function of training iterations
Problem 1: Most oracles generally extremely close
Problem 2: Performance on dev set not monotonically increasing as a function of training iterations

Solution: Tune number of iterations on dev data for each oracle
Problem 1: Most oracles generally extremely close
Problem 2: Performance on dev set not monotonically increasing as a function of training iterations

Solution: Tune number of iterations on dev data for each oracle

Final evaluation (test set): best static oracle vs best non-deterministic oracle
### Results – beam

<table>
<thead>
<tr>
<th>Language</th>
<th>Static</th>
<th>( \Delta ) non-det.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>85.05</td>
<td>+0.06</td>
</tr>
<tr>
<td>de</td>
<td>87.53</td>
<td>-0.23</td>
</tr>
<tr>
<td>en</td>
<td>90.35</td>
<td>+0.13</td>
</tr>
<tr>
<td>eu</td>
<td>79.97</td>
<td>+0.55</td>
</tr>
<tr>
<td>fr</td>
<td>83.10</td>
<td>-0.11</td>
</tr>
<tr>
<td>he</td>
<td>78.65</td>
<td>-0.39</td>
</tr>
<tr>
<td>hu</td>
<td>83.60</td>
<td>+0.08</td>
</tr>
<tr>
<td>ko</td>
<td>85.03</td>
<td>+0.09</td>
</tr>
<tr>
<td>pl</td>
<td>82.08</td>
<td>+1.26( ^\ddagger )</td>
</tr>
<tr>
<td>sv</td>
<td>79.05</td>
<td>-0.07</td>
</tr>
<tr>
<td><strong>Macro Avg.</strong></td>
<td><strong>83.59</strong></td>
<td><strong>0.14</strong></td>
</tr>
</tbody>
</table>
## Results – beam

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Δ non-det.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>85.05</td>
<td>+0.06</td>
</tr>
<tr>
<td>de</td>
<td>87.53</td>
<td>-0.23</td>
</tr>
<tr>
<td>en</td>
<td>90.35</td>
<td>+0.13</td>
</tr>
<tr>
<td>eu</td>
<td>79.97</td>
<td>+0.55</td>
</tr>
<tr>
<td>fr</td>
<td>83.10</td>
<td>-0.11</td>
</tr>
<tr>
<td>he</td>
<td>78.65</td>
<td>-0.39</td>
</tr>
<tr>
<td>hu</td>
<td>83.60</td>
<td>+0.08</td>
</tr>
<tr>
<td>ko</td>
<td>85.03</td>
<td>+0.09</td>
</tr>
<tr>
<td>pl</td>
<td>82.08</td>
<td>+1.26‡</td>
</tr>
<tr>
<td>sv</td>
<td>79.05</td>
<td>-0.07</td>
</tr>
<tr>
<td>Macro Avg.</td>
<td>83.59</td>
<td>0.14</td>
</tr>
<tr>
<td>Macro Avg. (w/o pl)</td>
<td>83.44</td>
<td>0.01</td>
</tr>
</tbody>
</table>

- Basically no difference, except Polish
## Results – greedy

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>Δ non-det.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>82.99</td>
<td>+0.04</td>
</tr>
<tr>
<td>de</td>
<td>84.22</td>
<td>+0.03</td>
</tr>
<tr>
<td>en</td>
<td>87.85</td>
<td>+0.60‡</td>
</tr>
<tr>
<td>eu</td>
<td>78.58</td>
<td>+0.24</td>
</tr>
<tr>
<td>fr</td>
<td>81.12</td>
<td>+0.40‡</td>
</tr>
<tr>
<td>he</td>
<td>75.27</td>
<td>+0.70†</td>
</tr>
<tr>
<td>hu</td>
<td>81.45</td>
<td>+0.22</td>
</tr>
<tr>
<td>ko</td>
<td>84.52</td>
<td>+0.30</td>
</tr>
<tr>
<td>pl</td>
<td>79.10</td>
<td>+1.33‡</td>
</tr>
<tr>
<td>sv</td>
<td>75.89</td>
<td>+0.39</td>
</tr>
<tr>
<td><strong>Macro Avg. (w/o pl)</strong></td>
<td><strong>82.39</strong></td>
<td><strong>+0.32</strong></td>
</tr>
</tbody>
</table>

- Without pl. not just zero
## Results – greedy

<table>
<thead>
<tr>
<th></th>
<th>Static</th>
<th>$\Delta$ non-det.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ar</td>
<td>82.99</td>
<td>+0.04</td>
</tr>
<tr>
<td>de</td>
<td>84.22</td>
<td>+0.03</td>
</tr>
<tr>
<td>en</td>
<td>87.85</td>
<td>+0.60‡</td>
</tr>
<tr>
<td>eu</td>
<td>78.58</td>
<td>+0.24</td>
</tr>
<tr>
<td>fr</td>
<td>81.12</td>
<td>+0.40‡</td>
</tr>
<tr>
<td>he</td>
<td>75.27</td>
<td>+0.70†</td>
</tr>
<tr>
<td>hu</td>
<td>81.45</td>
<td>+0.22</td>
</tr>
<tr>
<td>ko</td>
<td>84.52</td>
<td>+0.30</td>
</tr>
<tr>
<td>pl</td>
<td>79.10</td>
<td>+1.33‡</td>
</tr>
<tr>
<td>sv</td>
<td>75.89</td>
<td>+0.39</td>
</tr>
<tr>
<td>Macro Avg. (w/o pl)</td>
<td>82.39</td>
<td>+0.32</td>
</tr>
<tr>
<td>Macro Avg.</td>
<td>81.10</td>
<td>+0.43</td>
</tr>
</tbody>
</table>

- Without pl. not just zero
- Increases for all treebanks
Why does it only work with greedy? (speculative)

- Beam (search)
  - Search-based parsers are good at managing suboptimal local decisions (i.e., little error propagation)
  - No need to introduce additional ambiguity, search does the trick
Why does it only work with greedy? (speculative)

- **Beam (search)**
  - Search-based parsers are good at managing suboptimal local decisions (i.e., little error propagation)
  - No need to introduce additional ambiguity, search does the trick

- **Greedy**
  - Exposed to (some) more states during training, ⇒ generalizes better
  - Never harmful
Summary\textsuperscript{1}

- Spurious ambiguity in ArcStandard+Swap

\textsuperscript{1}Parser implementation available on my website http://www.ims.uni-stuttgart.de/~anders/
Summary

- Spurious ambiguity in ArcStandard+Swap
- Non-deterministic oracles

---

1Parser implementation available on my website
http://www.ims.uni-stuttgart.de/~anders/
Summary

- Spurious ambiguity in ArcStandard+Swap
- Non-deterministic oracles
- Parser accuracy
  - Beam: No improvement

1Parser implementation available on my website
http://www.ims.uni-stuttgart.de/~anders/
Summary

- Spurious ambiguity in ArcStandard+Swap
- Non-deterministic oracles
- Parser accuracy
  - **Beam**: No improvement
  - **Greedy**: Sometimes

---

1Parser implementation available on my website
http://www.ims.uni-stuttgart.de/~anders/
Questions

Thank you.

Questions?


Backup slide – Other ways of training (beam)

- Use early update, and update against the last correct item that fell off the beam
- Update against any gold sequence, pick the highest scoring (partial) one (may not coincide with best scoring complete sequence)

Moving target problem: across training iterations, correct sequence may change – more difficult to learn?
  - Train a model (with some oracle), apply it to the training data over the lattices and pick a single unique sequence for each sentence
  - Same as above, but do it with cross-validation (jack-knifing)

- All of these did worse than static oracle
Backup slide – Complexity of CanShift

- Theoretically $O(n^2)$

However, can stop if stack gets reduced to two tokens

In practice, marginal difference on overall training time
Backup slide – Complexity of CanShift

- Theoretically $O(n^2)$

- However, can stop if stack gets reduced to two tokens
Backup slide – Complexity of CanShift

- Theoretically $O(n^2)$
- However, can stop if stack gets reduced to two tokens
- In practice, marginal difference on overall training time