Training Deterministic Parsers Using Non-Deterministic Oracles

Joakim Nivre
Uppsala University
Department of Linguistics and Philology
joakim.nivre@lingfil.uu.se

Joint work with Yoav Goldberg, Bar-Ilan University
Introduction

Deterministic dependency parsing:
- Very fast: $10^5$ words per second
- Fairly accurate: 2–3% below the state of the art

How can we improve accuracy without losing speed?
Introduction

- Transition-based dependency parsing:
  - Define a transition system for dependency parsing
  - Train a classifier for predicting the next transition
  - Use the classifier to do deterministic parsing

- Current practice:
  - Train classifier on derivations produced by an oracle
  - Leads to error propagation at parsing time
  - Can be mitigated by using beam search – slowdown

- Novel idea:
  - Explore a larger search space during training
  - Allow the parser to make mistakes and recover
  - Requires a new type of oracle
Introduction

Outline

1. Transition-based dependency parsing
2. Old oracles – and why they are a problem
3. New oracles – and why they should help
4. Experiments
A dependency tree is a labeled directed tree $T$ with
- a set $V$ of nodes, labeled with words
- a set $A$ of arcs, labeled with dependency types

**Notation:**
- Arc $(w_i, d, w_j)$ links head $w_i$ to dependent $w_j$ with label $d$
- Shorthand: $w_i \xrightarrow{d} w_j \iff (w_i, d, w_j) \in A$

He sent her a letter.
Transition System: Configurations

- A parser configuration is a triple $c = (S, B, A)$, where
  - $S$ = a stack $[\ldots, w_i]_s$ of partially processed words,
  - $B$ = a buffer $[w_j, \ldots]_b$ of remaining input word,
  - $A$ = a set of labeled arcs $(w_i, d, w_j)$.

- Initialization:
  $([S], [w_1, \ldots, w_n]_b, \{\})$

- Termination:
  $(S, [\cdot]_b, A)$
Transition System: Transitions

Left-Arc\((d)\): 
\[
\frac{([\ldots, w_i]_S, [w_j, \ldots]_B, A)}{([\ldots]_S, [w_j, \ldots]_B, A \cup \{(w_j, d, w_i)\})} \quad \neg \text{HEAD}(w_i)
\]

Right-Arc\((d)\): 
\[
\frac{([\ldots, w_i]_S, [w_j, \ldots]_B, A)}{([\ldots, w_i, w_j]_S, [\ldots]_B, A \cup \{(w_i, d, w_j)\})}
\]

Reduce: 
\[
\frac{([\ldots, w_i]_S, B, A)}{([\ldots]_S, B, A)} \quad \text{HEAD}(w_i)
\]

Shift: 
\[
\frac{([\ldots]_S, [w_i, \ldots]_B, A)}{([\ldots, w_i]_S, [\ldots]_B, A)}
\]
Parse Example

Transitions:

Stack

Buffer

Arcs

[ ] \text{S}

[He, sent, her, a, letter, .] \text{B}

He \quad sent \quad her \quad a \quad letter \quad .
Parse Example

Transitions: SH

Stack          Buffer          Arcs

[He]_S          [sent, her, a, letter, .]_B

He    sent    her    a     letter    .

Training Deterministic Parsers Using Non-Deterministic Oracles
Parse Example

Transitions: SH-LA

Stack | Buffer | Arcs
--- | --- | ---
[ ]$S$ | [sent, her, a, letter, .]$B$ | He $\xrightarrow{SBJ}$ sent
Parse Example

**Transitions:** SH-LA-SH

**Stack**

\[ \text{sent}_S \]

**Buffer**

\[ \text{her, a, letter, .}_B \]

**Arcs**

He $\xleftarrow{\text{SBJ}}$ sent
Parse Example

Transitions: SH-LA-SH-RA

Stack
\[ [\text{sent, her}]_S \]

Buffer
\[ [\text{a, letter, .}]_B \]

Arcs
\[ \text{He} \xleftarrow{\text{SBJ}} \text{sent} \xrightarrow{\text{IOBJ}} \text{her} \]

He sent her a letter .
Parse Example

Transitions: SH-LA-SH-RA-SH

Stack | Buffer | Arrows
--- | --- | ---
[sent, her, a]_S | [letter, .]_B | He \(\textit{SBJ}\) sent \\
| | | sent \(\textit{IOBJ}\) her

He sent her a letter .
Parse Example

Transitions: SH-LA-SH-RA-SH-LA

Stack: [sent, her]$_S$
Buffer: [letter, .]$_B$

Arcs:
He $\xleftarrow{\text{SBJ}}$ sent
sent $\xrightarrow{\text{IOBJ}}$ her
a $\xleftarrow{\text{DET}}$ letter

He sent her a letter.
Parse Example

**Transitions:** SH-LA-SH-RA-SH-LA-RE

**Stack**

[sent]$_S$

**Buffer**

[letter, .]$_B$

**Arcs**

He $\leftarrow$ SBJ sent

sent $\rightarrow$ IOBJ her

a $\leftarrow$ DET letter

He sent her a letter .
Parse Example

Transitions: SH-LA-SH-RA-SH-LA-RE-RA

Stack       Buffer          Arcs
[ sent, letter ]_S  [ . ]_B  He SBJ sent
                    sent IOBJ her
                    a DET letter
                    sent DOBJ letter
Parse Example


Stack Buffer Arcs

[sent]_S [.]_B

He sent her a letter .

He SBJ sent IOBJ her DET a DET sent DOBJ letter

Training Deterministic Parsers Using Non-Deterministic Oracles 8(23)
Parse Example


Stack
[sent, .]_S

Buffer
[ ]_B

Arcs
He \(\xleftrightarrow{SBJ}\) sent
sent \(\xrightarrow{IOBJ}\) her
a \(\xleftrightarrow{DET}\) letter
sent \(\xrightarrow{DOBJ}\) letter
sent \(\xrightarrow{PUNC}\).
To guide the parser we use a (linear) classifier:

\[ t^* = \arg\max_{t} \mathbf{w} \cdot f(c, t) \]

- History-based feature representation \( f(c, t) \):
  - Features over input tokens relative to \( S \) and \( B \)
  - Features over the (partial) dependency tree defined by \( A \)
  - Features over the (partial) transition sequence

- Weight vector \( \mathbf{w} \) learned from treebank data
Deterministic Parsing

\[
\text{PARSE}(w_1, \ldots, w_n, \mathbf{w}) \\
1 \quad c \leftarrow ([S, [w_1, \ldots, w_n]_{B}, \{\}]) \\
2 \quad \textbf{while } B_c \neq [] \\
3 \quad t^* \leftarrow \text{argmax}_t \mathbf{w} \cdot f(c, t) \\
4 \quad c \leftarrow t^*(c) \\
5 \quad \textbf{return } T = (\{w_1, \ldots, w_n\}, A_c)
\]
Online Learning with an Oracle

\[
\text{LEARN}(\{T_1, \ldots, T_N\})
\]

1. \( w \leftarrow 0.0 \)
2. \( \text{for } i \text{ in } 1..K \)
3. \( \quad \text{for } j \text{ in } 1..N \)
4. \( \quad c \leftarrow ([], [w_1, \ldots, w_{n_j}]_B, \{\}) \)
5. \( \quad \textbf{while } B_c \neq [] \)
6. \( \quad t^* \leftarrow \arg\max_{t} w \cdot f(c, t) \)
7. \( \quad t_o \leftarrow o(c, T_i) \)
8. \( \quad \textbf{if } t^* \neq t_o \)
9. \( \quad w \leftarrow w + f(c, t_o) - f(c, t^*) \)
10. \( \quad c \leftarrow t_o(c) \)
11. \( \text{return } w \)
Online Learning with an Oracle

\begin{algorithm}
\textbf{LEARN}(\{T_1, \ldots, T_N\})
\begin{algorithmic}
\State $w \leftarrow 0.0$
\For{$i$ in 1..$K$}
\For{$j$ in 1..$N$}
\State $c \leftarrow ([], [w_1, \ldots, w_n]_B, \{\})$
\While{$B_c \neq []$}
\State $t^* \leftarrow \text{argmax}_t w \cdot f(c, t)$
\State $t_o \leftarrow o(c, T_i)$
\If{$t^* \neq t_o$}
\State $w \leftarrow w + f(c, t_o) - f(c, t^*)$
\EndIf
\EndWhile
\EndFor
\EndFor
\State $c \leftarrow t_o(c)$
\State \textbf{return} $w$
\end{algorithmic}
\end{algorithm}

- Oracle $o(c, T_i)$ returns the optimal transition for $c$ and $T_i$
Standard Oracle for Arc-Eager Parsing

\[ o(c, T) = \begin{cases} 
\text{Left-Arc} & \text{if } \text{top}(S_c) \leftarrow \text{first}(B_c) \text{ in } T \\
\text{Right-Arc} & \text{if } \text{top}(S_c) \rightarrow \text{first}(B_c) \text{ in } T \\
\text{Reduce} & \text{if } \exists w < \text{top}(S_c) : w \leftrightarrow \text{first}(B_c) \text{ in } T \\
\text{Shift} & \text{otherwise} 
\end{cases} \]

- **Correct:**
  - Derives \( T \) in a configuration sequence \( C(o, T) = c_0, \ldots, c_m \)

- **Problems:**
  - Deterministic: Ignores other derivations of \( T \)
  - Incomplete: Valid only for configurations in \( C(o, T) \)
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

Stack
[ ]

Buffer
[He, sent, her, a, letter, .]_B

Arcs

He, sent, her, a, letter, .
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH

Stack
[He]_S

Buffer
[sent, her, a, letter, .]_B

Arcs

He    sent    her    a     letter    .

Training Deterministic Parsers Using Non-Deterministic Oracles
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA

Stack
[ ]

Buffer
[sent, her, a, letter, .]_B

Arcs
He ←^SBJ^ sent

He  sent  her  a  letter  .
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH

Stack
[ sent ]_S

Buffer
[ her, a, letter, . ]_B

Arcs
He ⇐ sent

He    sent    her    a     letter    .
Non-Determinism

**Transitions:**

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA

**Stack**

\( [\text{sent, her}]_S \)

**Buffer**

\( [\text{a, letter, .}]_B \)

**Arcs**

He \( \xleftarrow{\text{SBJ}} \) sent

sent \( \xrightarrow{\text{IOBJ}} \) her

He sent her a letter .
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE

Stack

[\text{sent}]_S

Buffer

[a, \text{ letter, .}]_B

Arcs

He \xleftarrow{\text{SBJ}} \text{ sent}

\text{sent} \xrightarrow{\text{IOBJ}} \text{ her}

He \hspace{1cm} \text{ sent} \hspace{1cm} \text{ her} \hspace{1cm} \text{ a} \hspace{1cm} \text{ letter} \hspace{1cm} .
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH

Stack
[ sent, a ]

Buffer
[ letter, . ]

Arcs
He \( \xrightarrow{\text{SBJ}} \) sent
sent \( \xrightarrow{\text{IOBJ}} \) her

He sent her a letter .
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH-LA

Stack

[send]_S

Buffer

[letter, .]_B

Arcs

He $\leftarrow_{SBJ}$ sent
sent $\rightarrow_{IOBJ}$ her
a $\leftarrow_{DET}$ letter

He sent her a letter.
Non-Determinism

Transitions:
SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH-LA-RA

Stack
[send, letter]_S

Buffer
[.}_B

Arcs
He \leftrightarrow_{SBJ} sent
sent \leftrightarrow_{IOBJ} her
a \leftrightarrow_{DET} letter
sent \leftrightarrow_{DOBJ} letter
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA

SH-LA-SH-RA-RE-SH-LA-RA-RE

Stack

[sent]_S

Buffer

[.]_B

Arcs

He \leftarrow^{SBJ} \text{sent}

sent \rightarrow^{IOBJ} \text{her}

a \leftarrow^{DET} \text{letter}

sent \rightarrow^{DOBJ} \text{letter}

He \hspace{1cm} \text{sent} \hspace{1cm} \text{her} \hspace{1cm} a \hspace{1cm} \text{letter} \hspace{1cm} .
Non-Determinism

Transitions:

SH-LA-SH-RA-SH-LA-RE-RA-RE-RA
SH-LA-SH-RA-RE-SH-LA-RA-RE-RA

Stack

[sent, .]_{S}

Buffer

[ ]_{B}

Arcs

He $\leftarrow^{SBJ}$ sent
sent $\rightarrow^{IOBJ}$ her
a $\leftarrow^{DET}$ letter
sent $\rightarrow^{DOBJ}$ letter
sent $\rightarrow^{PUNC}$ .
Non-Optimality

Transitions:

Stack

Buffer

Arcs

\[ \text{Sent, her, a, letter, .} ]_B

\[ \text{He, sent, her, a, letter, .} ]

He \quad sent \quad her \quad a \quad letter \quad .
Non-Optimality

Transitions:

Stack: [He]$_S$

Buffer: [sent, her, a, letter, .]$_B$

Arcs

He sent her a letter .
Non-Optimality

Transitions: SH-LA

Stack

Buffer

Arcs

He sent her a letter .

He sent her a letter .
Non-Optimality

Transitions: SH-LA-SH

Stack: [sent]$_S$
Buffer: [her, a, letter, .]$_B$

Arcs: He $\leftarrow$ sent

He sent her a letter.
Non-Optimality

Transitions: \( \text{SH-LA-SH-SH} \)

Stack

\([\text{sent, her}]_S\)

Buffer

\([a, \text{ letter, .}]_B\)

Arcs

He \(\overleftarrow{\text{SBJ}}\) sent

He \(\overrightarrow{\text{SBJ}}\) sent her a letter .

Training Deterministic Parsers Using Non-Deterministic Oracles
Non-Optimality

Transitions:

SH-LA-SH-SH-SH

Stack

[ sent, her, a ]_S

Buffer

[ letter, . ]_B

Arcs

He ↕ sent
Non-Optimality

Transitions:

Stack

\([\text{sent, her}]_S\)

Buffer

\([\text{letter, .}]_B\)

Arcs

\(\text{He} \xrightarrow{\text{SBJ}} \text{sent} \xleftarrow{\text{DET}} \text{a} \xrightarrow{\text{DET}} \text{letter} \)

\(\text{He sent her a letter .}\)
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-SH

Stack

[ sent, her, letter ] S

Buffer

[ . ] B

Arcs

He \( \text{SBJ} \) sent

a \( \text{DET} \) letter
Non-Optimality

Transitions:


Stack

[sent, her, letter, .]_S

Buffer

[ ]_B

Arcs

He SBJ sent
a DET letter

He sent her a letter .
Non-Optimality

Transitions:


Stack

[ ]s

Buffer

[He, sent, her, a, letter, .]B

Arcs

PUNC

DOBJ

IOBJ

DET

SBJ

He sent her a letter .
Non-Optimality

Transitions:


Stack

[He]_S

Buffer

[sent, her, a, letter, .]_B

Arcs

He    sent    her    a     letter    .

SBJ

DOBJ

IOBJ

DET

PUNC
Non-Optimality

Transitions:

SH-LA

Stack
[ ]_{s}

Buffer
[ sent, her, a, letter, . ]_{B}

Arcs
He $^{SBJ}$ sent

He sent her a letter .
Non-Optimality

Transitions:
SH-LA-SH

Stack
[sent]$_S$

Buffer
[her, a, letter, .]$_B$

Arcs
He $\xleftarrow{\text{SBJ}}$ sent

---

He sent her a letter.
Non-Optimality

Transitions:

SH-LA-SH-SH

Stack

[sent, her]_S

Buffer

[a, letter, .]_B

Arcs

He \( \leftarrow \) sent

Training Deterministic Parsers Using Non-Deterministic Oracles
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-SH-LA-SH-SH

SH-LA-SH-SH-SH

Stack

[sent, her, a]_S

Buffer

[letter, .]_B

Arcs

He \(\xleftarrow{\text{SBJ}}\) sent

He sent her a letter .

Training Deterministic Parsers Using Non-Deterministic Oracles 14(23)
Non-Optimality

Transitions:
- SH-LA-SH-SH-SH-LA

Stack
- [sent, her]$_S$

Buffer
- [letter, .]$_B$

Arcs
- He $\underset{\text{SBJ}}{\leftarrow}$ sent
- a $\underset{\text{DET}}{\leftarrow}$ letter
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-LA

Stack

[ sent ] _S

Buffer

[ letter, . ] _B

Arcs

He \( \xleftarrow{SBJ} \) sent
a \( \xleftarrow{DET} \) letter
her \( \xleftarrow{x} \) letter

He sent her a letter .
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-LA-RA

Stack

[.][_B]

Buffer

He sent her a letter.

Arcs

He \(\text{SBJ}\) sent a \(\text{DET}\) letter
her \(\text{X}\) letter
sent \(\text{DOBJ}\) letter
Non-Optimality

Transitions:

SH-LA-SH-SH-SH-LA-SH-SH
SH-LA-SH-SH-SH-LA-LA-RA-RE

Stack

[sent]_S

Buffer

[.]_B

Arcs

He \text{ SBJ } sent
a \text{ DET } letter
her \text{ X } letter
sent \text{ DOBJ } letter
Non-Optimality

Transitions:


Stack
[sent, .]_{S}

Buffer
[ ]_{B}

Arcs
He $\stackrel{SBJ}{\leftarrow}$ sent
a $\stackrel{DET}{\leftrightarrow}$ letter
her $\stackrel{x}{\leftarrow}$ letter
sent $\stackrel{DOBJ}{\rightarrow}$ letter
sent $\stackrel{PUNC}{\rightarrow}$. 

He sent her a letter.
Rethinking Oracles

- New idea:
  - A transition is optimal if the best tree remains reachable
  - Best tree = argmin$_{T, T'}$ $L(T, T')$

- New view of oracle:
  - Boolean function $o(c, t, T) = \text{true}$ if $t$ is optimal for $c$ and $T$
  - Non-deterministic: More than one transition can be optimal
  - Complete: Correct for all configurations

- New problem:
  - How do we know which trees are reachable?
Reachability for Arcs and Trees

- **Arc reachability:**
  - An arc \( w_i \rightarrow w_j \) is reachable in \( c \) iff \( w_i \rightarrow w_j \in A_c \), or \( w_i \in S_c \cup B_c \) and \( w_j \in B_c \) (same for \( w_i \leftarrow w_j \))

- **Tree reachability:**
  - A (projective) tree \( T \) is reachable in \( c \) iff every arc in \( T \) is reachable in \( c \)

- **Arc-decomposable system:**
  - Tree reachability reduces to arc reachability
  - Holds for some transition systems but not all
A New Oracle

\[ \mathcal{R}(c) \equiv \{ a \mid a \text{ is an arc reachable in } c \} \]

\[ o(c, t, T) = \begin{cases} 
\text{true} & \text{if } [\mathcal{R}(c) - \mathcal{R}(t(c))] \cap T = \emptyset \\
\text{false} & \text{otherwise} 
\end{cases} \]
Case by Case

- Notation: $s =$ word on top of stack, $b =$ first word in buffer

\[
\begin{align*}
o(c, \text{LA}, T) &= \begin{cases} 
\text{false} & \text{if } \exists w \in B_c : s \leftrightarrow w \in T \text{ (except } s \leftarrow b) \\
\text{true} & \text{otherwise}
\end{cases} \\
o(c, \text{RA}, T) &= \begin{cases} 
\text{false} & \text{if } \exists w \in S_c : w \leftrightarrow b \in T \text{ (except } s \rightarrow b) \\
\text{true} & \text{otherwise}
\end{cases} \\
o(c, \text{RE}, T) &= \begin{cases} 
\text{false} & \text{if } \exists w \in B_c : s \rightarrow w \in T \\
\text{true} & \text{otherwise}
\end{cases} \\
o(c, \text{SH}, T) &= \begin{cases} 
\text{false} & \text{if } \exists w \in S_c : w \leftrightarrow b \in T \\
\text{true} & \text{otherwise}
\end{cases}
\end{align*}
\]
A New Learning Algorithm

\textsc{Learn}(\{T_1, \ldots, T_N\})

1. \textbf{w} \leftarrow 0.0
2. \textbf{for} \ i \ \textbf{in} \ 1..K
3. \textbf{for} \ j \ \textbf{in} \ 1..N
4. \hspace{1cm} c \leftarrow ([ ]_s, [w_1, \ldots, w_n]_B, \{ \})
5. \hspace{1cm} \textbf{while} \ B_c \neq []
6. \hspace{2cm} t^* \leftarrow \text{argmax}_t \textbf{w} \cdot \textbf{f}(c, t)
7. \hspace{1cm} t_o \leftarrow \text{argmax}_{t \in \{ t \mid o(c, t, T_i) \}} \textbf{w} \cdot \textbf{f}(c, t)
8. \hspace{1cm} \textbf{if} \ t^* \neq t_o
9. \hspace{2cm} \textbf{w} \leftarrow \textbf{w} + \textbf{f}(c, t_o) - \textbf{f}(c, t^*)
10. \hspace{1cm} c \leftarrow \text{CHOICE}(t_o(c), t^*(c))
11. \textbf{return} \textbf{w}
A New Learning Algorithm

\textbf{LEARN}(\{T_1, \ldots, T_N\})

1. \( w \leftarrow 0.0 \)
2. \textbf{for} \( i \) in 1..\( K \)
3. \hspace{1em} \textbf{for} \( j \) in 1..\( N \)
4. \hspace{2em} \( c \leftarrow ([], [w_1, \ldots, w_{n_j}]_B, \{\}) \)
5. \hspace{1em} \textbf{while} \( B_c \neq [] \)
6. \hspace{2em} \( t^* \leftarrow \arg\max_t w \cdot f(c, t) \)
7. \hspace{2em} \( t_o \leftarrow \arg\max_{t \in \{t | o(c, t, T_i)\}} w \cdot f(c, t) \)
8. \hspace{2em} \textbf{if} \( t^* \neq t_o \)
9. \hspace{3em} \( w \leftarrow w + f(c, t_o) - f(c, t^*) \)
10. \hspace{1em} \( c \leftarrow \text{CHOICE}(t_o(c), t^*(c)) \)
11. \textbf{return} \( w \)
Experimental Evaluation

▶ Data sets:
  ▶ English treebanks: WSJ, Brown, BNC, Google Web
  ▶ Multilingual: CoNLL 2007 Shared Task

▶ Settings:
  ▶ Greedy: Old learning algorithm
  ▶ Greedy + Ambiguity: $\text{CHOICE}(t_o(c), t^*(c)) = t_o(c)$
  ▶ Greedy + Exploration: Random $\text{CHOICE}(t_o(c), t^*(c))$
English Results

Experiments
Experiments

Multilingual Results

<table>
<thead>
<tr>
<th>Language</th>
<th>Greedy</th>
<th>Greedy + Ambiguity</th>
<th>Greedy + Exploration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Catalan</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chinese</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hungarian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Italian</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parsing Accuracy
Conclusion

- Exploring a larger search space at training time helps
  - Allowing non-canonical derivations (spurious ambiguity)
  - Learning optimal transitions in non-optimal configurations
- Requires a new type of oracle
  - Non-deterministic: more than one transition may be optimal
  - Complete: optimality defined for all configurations
- Parsing remains deterministic (fast)