“Training Deterministic Parsers with Non-Deterministic Oracles”
by Yoav Goldberg and Joakim Nivre, 2013

Seminarvortrag
Pius Meinert
July 13, 2018
Training Deterministic Parsers with Non-Deterministic Oracles

He_1 wrote_2 her_3 a_4 letter_5
Training **Deterministic Parsers** with Non-Deterministic Oracles
Definition (Transition System)

A transition system for dependency parsing is a quadruple $S = (C, T, c_s, C_t)$, where

1. $C$ is a set (configurations),
2. $T$ is a set of transitions, each of which is a (partial) function $t : C \rightarrow C$,
3. $c_s$ is an initialization function, mapping sentence $w = w_1w_2...w_n$ to a configuration $c \in C$,
4. $C_t \subseteq C$ (terminal configurations).
He 1 wrote 2 her 3 a 4 letter 5
Training **Deterministic Parsers** with Non-Deterministic Oracles

\[
\text{ROOT} \quad \text{He}_1 \quad \text{wrote}_2 \quad \text{her}_3 \quad \text{a}_4 \quad \text{letter}_5
\]

\[
\big| c_s(w) \big| \quad \text{[ROOT], [He}_1, \text{wrote}_2, \text{her}_3, \text{a}_4, \text{letter}_5], \{\}\big]
\]
Training **Deterministic Parsers** with Non-Deterministic Oracles

```
ROOT   He₁   wrote₂   her₃   a₄   letter₅

[ROOT], [He₁, wrote₂, her₃, a₄, letter₅]

| SHIFT |
[ROOT, He₁], [wrote₂, her₃, a₄, letter₅]
```
Training **Deterministic Parsers** with Non-Deterministic Oracles

He wrote her a letter

\[ \text{[ROOT, He},\text{1]}], [\text{wrote}_2, \text{her}_3, \text{a}_4, \text{letter}_5] \]

LEFT\text{SBJ}

\[ \text{[ROOT]}, [\text{wrote}_2, \text{her}_3, \text{a}_4, \text{letter}_5] \]
Training **Deterministic Parsers** with Non-Deterministic Oracles
Training **Deterministic Parsers** with Non-Deterministic Oracles

\[
\begin{align*}
\text{[ROOT, wrote}_2\text{], [her}_3\text{, a}_4\text{, letter}_5\text{]} \\
\text{\textup{RIGHT}}_{\text{IOBJ}}, \text{\textup{SHIFT}}, \text{\textup{LEFT}}_{\text{DET}}, \text{\textup{REDUCE}}, \text{\textup{RIGHT}}_{\text{DOBJ}} \\
\text{[ROOT, wrote}_2\text{, letter}_5\text{], [ ] } \in C_t
\end{align*}
\]
Training Deterministic Parsers with Non-Deterministic Oracles

1. if \( c = (\sigma| i, j| \beta, A) \) and \((j, i) \in T\) then
2. \( t \leftarrow \text{LEFT} \)
3. else if \( c = (\sigma| i, j| \beta, A) \) and \((i, j) \in T\) then
4. \( t \leftarrow \text{RIGHT} \)
5. else if \( c = (\sigma| i, j| \beta, A) \) and \( \exists k [k < i \land [(k, j) \in T \lor (j, k) \in T]] \) then
6. \( t \leftarrow \text{REDUCE} \)
7. else
8. \( t \leftarrow \text{SHIFT} \)
9. return \( t \)
Greedy Classifier-based Parsing

\[ c \leftarrow c_s(w) \]

\[ \textbf{while } c \notin C_t \textbf{ do} \]

\[ t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} w \cdot \phi(c, t) \]

\[ c \leftarrow t_p(c) \]

\[ \textbf{return } A_c \]
for \((w, T) \in d\) do

\[ c \leftarrow c_s(w) \]

while \(c \notin C_t\) do

\[ t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} w \cdot \phi(c, t) \]

\[ \text{CORRECT}(c) \leftarrow \{ t \mid o(t; c, T) = \text{true} \} \]

\[ t_o \leftarrow \arg \max_{t \in \text{CORRECT}(c)} w \cdot \phi(c, t) \]

if \(t_p \notin \text{CORRECT}(c)\) then

\[ \text{UPDATE}(w, \phi(c, t_o), \phi(c, t_p)) \]

\[ c \leftarrow t_o(c) \]

else

\[ c \leftarrow t_p(c) \]

return \(w\)
Training Deterministic Parsers with Non-Deterministic Oracles

He wrote her a letter

PRD
SBJ
DOBJ
IOBJ
DET
SH, LA_{SBJ}, RA_{PRD}, RA_{IOBJ}, SH, LA_{DET}, RE, RA_{DOBJ}
Training Deterministic Parsers with Non-Deterministic Oracles

SPR, LA_{SBJ}, RA_{PRD}, RA_{IOBJ}, SH, LA_{DET}, RE, RA_{DOBJ}

SH, LA_{SBJ}, RA_{PRD}, RA_{IOBJ}, RE, SH, LA_{DET}, RA_{DOBJ}

→ spurious ambiguity requires non-deterministic oracle instead of static oracle
... with Non-Deterministic and Complete Oracles

\[ \text{[ROOT], [He}_1, \text{wrote}_2, \text{her}_3, \text{a}_4, \text{letter}_5] \]

\[ \text{SH, LA}_{\text{SBJ}}, \text{RA}_{\text{PRD}}, \text{SH} \]

\[ \text{[ROOT, wrote}_2, \text{her}_3], \text{[a}_4, \text{letter}_5] \]
... with Non-Deterministic and Complete Oracles

\[
\text{[ROOT, wrote}_2, \text{her}_3], [a_4, \text{letter}_5] \in C_t
\]

\[ \uparrow \quad \text{SH, LA}_{\text{DET}}, \text{SH} \]

→ error propagation can be mitigated by complete oracle

→ **dynamic oracle**: non-deterministic + complete
Training (Standard)

for \((w, T) \in d\) do
  \(c \leftarrow c_s(w)\)
  while \(c \notin C_t\) do
    \(t_p \leftarrow \arg\max_{t \in \text{LEGAL}(c)} w \cdot \phi(c, t)\)
    \(\text{CORRECT}(c) \leftarrow \{t \mid o(t; c, T) = \text{true}\}\)
    \(t_o \leftarrow \arg\max_{t \in \text{CORRECT}(c)} w \cdot \phi(c, t)\)
    if \(t_p \notin \text{CORRECT}(c)\) then
      UPDATE\((w, \phi(c, t_o), \phi(c, t_p))\)
      \(c \leftarrow t_o(c)\)
    else
      \(c \leftarrow t_p(c)\)
  return \(w\)
Training with Exploration

\begin{algorithm}
\begin{algorithmic}[1]
\State \For{$(w, T) \in d$} \Do
\State $c \leftarrow c_s(w)$
\State While $c \notin C_t$ \Do
\State $t_p \leftarrow \arg \max_{t \in \text{LEGAL}(c)} \ w \cdot \phi(c, t)$
\State $\text{OPTIMAL}(c) \leftarrow \{t \mid o(t; c, T) = \text{true}\}$
\State $t_o \leftarrow \arg \max_{t \in \text{OPTIMAL}(c)} \ w \cdot \phi(c, t)$
\State \If{$t_p \notin \text{OPTIMAL}(c)$}
\State \quad \text{UPDATE}(w, \phi(c, t_o), \phi(c, t_p))$
\State \quad $c \leftarrow \text{EXPLORE}(c, t_o, t_p)$
\State \Else
\State \quad $c \leftarrow t_p(c)$
\State \EndIf
\State \EndWhile
\State \EndFor
\State \Return $w$
\end{algorithmic}
\end{algorithm}
He wrote her a letter.

\[ \text{PRD} \rightarrow \text{SBJ} \]

\[ \text{ROOT} \rightarrow \text{He} \rightarrow \text{wrote} \rightarrow \text{her} \rightarrow \text{a} \rightarrow \text{letter} \]
He wrote her a letter.

\[ C(A, T) = 2 \]
He wrote her a letter.

\[ \text{ROOT, wrote}_2, \text{her}_3, [a_4, \text{letter}_5] \]
He wrote her a letter.

\[
\min_{A:C \Rightarrow A} C(A, T) = 0
\]
He wrote her a letter.

\[ \text{PRD, SBJ, ROOT, wrote, her, a, letter} \]

\[ \text{[ROOT, wrote}, 2, \text{her}, 3, \text{a}, 4, \text{letter} 5] \]

\[ \text{SH, ...} \]
Optimality / Transition Costs

\[ C(\text{SHIFT}; c, T) = \min_{A:t(c) \rightarrow A} C(A, T) - \min_{A:c \rightarrow A} C(A, T) = 1 \]
$$C(\text{SHIFT}; c, T) = \min_{A:t(c)\sim A} C(A, T) - \min_{A:c\sim A} C(A, T) = 1$$

$$o_d(c, T) = \{t \mid C(t; c, T) = 0\}$$
### Definition (Tree Consistency)

A set of arcs $A$ is said to be *tree consistent* if there exists a projective dependency tree $T$ such that $A \subseteq T$.

### Definition (Arc Decomposition)

A transition system is said to be *arc decomposable* if, for every tree consistent arc set $A$ and configuration $c$, $c \rightsquigarrow A$ is entailed by $c \rightsquigarrow (h, d)$ for every arc $(h, d) \in A$. 
Arc Decomposition - Arc-Standard Counterexample

\[ c = ([a, b, c], \beta) \]

Arc-Standard Transitions

\[
\text{LEFT}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})
\]

\[
\text{RIGHT}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})
\]

\[
\text{SHIFT}[(\sigma, b|\beta, A)] = (\sigma|b, \beta, A)
\]
Arc Decomposition - Arc-Standard Counterexample

\[ c = ([a, b, c], \beta) \vdash ([a, c], \beta) \]

Arc-Standard Transitions

\[
\begin{align*}
\text{LEFT}[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\}) \\
\text{RIGHT}[(\sigma|s_1|s_0, \beta, A)] &= (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\}) \\
\text{SHIFT}[(\sigma, b|\beta, A)] &= (\sigma|b, \beta, A)
\end{align*}
\]
c = ([a, b, c], \beta) \vdash ([a, b], \beta) \vdash ([b], \beta)

Arc-Standard Transitions
\text{LEFT}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})
\text{RIGHT}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})
\text{SHIFT}[(\sigma, b|\beta, A)] = (\sigma|b, \beta, A)
**Given:** arbitrary configuration $c = (\sigma, \beta, A)$ and tree consistent arc set $A'$ such that all arc are reachable from $c$.

**To show:** $c \Rightarrow A'$

\[
\overline{B} = \{(h, d) \mid h, d \not\in \beta\} \\
B = \{(h, d) \mid h, d \in \beta\} \\
B_h = \{(h, d) \mid h \in \beta, d \in \sigma\} \\
B_d = \{(h, d) \mid d \in \beta, h \in \sigma\}
\]
\[ \bar{\mathcal{B}} = \{(h, d) \mid h, d \notin \beta\} \]
\[ \mathcal{B} = \{(h, d) \mid h, d \in \beta\} \]
\[ \mathcal{B}_h = \{(h, d) \mid h \in \beta, d \in \sigma\} \]
\[ \mathcal{B}_d = \{(h, d) \mid d \in \beta, h \in \sigma\} \]
Arc Decomposition - Arc-Eager Proof Sketch

$$\overline{\mathcal{B}} = \{(h, d) \mid h, d \notin \beta\}$$

$$\mathcal{B} = \{(h, d) \mid h, d \in \beta\}$$

$$\mathcal{B}_h = \{(h, d) \mid h \in \beta, d \in \sigma\}$$

$$\mathcal{B}_d = \{(h, d) \mid d \in \beta, h \in \sigma\}$$
Arc Decomposition - Arc-Eager Proof Sketch

\[ B = \{(h, d) \mid h, d \in \beta\} \]

\[ \bar{B} = \{(h, d) \mid h, d \notin \beta\} \]

\[ B_h = \{(h, d) \mid h \in \beta, d \in \sigma\} \]

\[ B_d = \{(h, d) \mid d \in \beta, h \in \sigma\} \]
\[ \overline{B} = \{(h, d) \mid h, d \notin \beta\} \]
\[ B = \{(h, d) \mid h, d \in \beta\} \]
\[ B_h = \{(h, d) \mid h \in \beta, d \in \sigma\} \]
\[ B_d = \{(h, d) \mid d \in \beta, h \in \sigma\} \]
Arc Decomposition - Arc-Eager Proof Sketch

\[ \overline{B} = \{(h, d) \mid h, d \notin \beta\} \]
\[ B = \{(h, d) \mid h, d \in \beta\} \]
\[ B_h = \{(h, d) \mid h \in \beta, d \in \sigma\} \]
\[ B_d = \{(h, d) \mid d \in \beta, h \in \sigma\} \]
Dynamic Oracle

\[ o_d(c, T) = \{ t \mid C(t; c, T) = 0 \} \]
\[ C(t; c, T) = \min_{A:t(c)\rightarrow A} C(A, T) - \min_{A:c\rightarrow A} C(A, T) \]

**Efficiently compute transition costs:**

1. Intersect set of individually reachable arcs with goal arc set.
2. Gain set of individually reachable goal arcs and thusly, reachable goal arc set.
3. See how a given transition affects this set of reachable arcs.
Transition Systems

✓ Arc-Eager
  • Nivre 2003
  • Goldberg and Nivre 2012

✖ Arc-Standard
  • Nivre 2004
  • Goldberg, Sartorio, and Satta 2014

✓ Hybrid
  • Kuhlmann, Gómez-Rodríguez, and Satta 2011

✓ Easy-First
  • Goldberg and Elhadad 2010
Conclusion

- spurious ambiguity
  static $\rightarrow$ non-deterministic oracle
- error propagation
  incomplete $\rightarrow$ complete oracle
- **dynamic oracle** (non-deterministic + complete)
- arc decomposability
- good runtime
  optimization during training
- experiments show improved accuracy


Transition Costs - Arc-Eager

\[ C(\text{SHIFT}; (\sigma, b|\beta, A), T) = |\{(k, b) \in T \mid k \in \sigma\} \]
\[ \quad \cup \{ (b, k) \in T \mid k \in \sigma \land \forall x \in V : (x, k) \notin A \} | \]

\[ C(\text{RIGHT}; (\sigma|s, b|\beta, A), T) = |\{(k, b) \in T \mid k \in \sigma \cup \beta\} \]
\[ \quad \cup \{ (b, k) \in T \mid k \in \sigma \land \forall x \in V : (x, k) \notin A \} | \]

\[ C(\text{LEFT}; (\sigma|s, b|\beta, A), T) = |\{(k, s) \in T \mid k \in \beta\} \cup \{ (s, k) \in T \mid k \in \beta\} | \]

\[ C(\text{REDUCE}; (\sigma|s, \beta, A), T) = |\{(s, k) \in T \mid k \in \beta\}| \]
Transition Costs - Hybrid

\[ C(\text{SHIFT}; (\sigma|s_1|s_0, b|\beta, A), T) = |\{(b, k) \in T | k \in \{s_0, s_1\} \cup \sigma\} \]
\[ \quad \quad \cup \{(k, b) \in T | k \in \{s_1\} \cup \sigma\}| \]

\[ C(\text{RIGHT}; (\sigma|s_1|s_0, \beta, A), T) = |\{(s_0, k) \in T | k \in \beta\} \]
\[ \quad \quad \cup \{(k, s_0) \in T | k \in \beta\}| \]

\[ C(\text{LEFT}; (\sigma|s_1|s_0, b|\beta, A), T) = |\{(s_0, k) \in T | k \in \{b\} \cup \beta\} \]
\[ \quad \quad \cup \{(k, s_0) \in T | k \in \{s_1\} \cup \beta\}| \]
Transition Costs - Easy-First

\[
C(\text{TR}; (\lambda, A), T) = |\{(h', d) \in T \mid h' \in \lambda \land h' \neq h\} \cup \{(d, d') \in T \mid d' \in \lambda\}| \\
\text{TR} \in \{\text{LEFT}_{lb}^i \mid 1 < i \leq |\lambda|\} \cup \{\text{RIGHT}_{lb}^i \mid 1 \leq i < |\lambda|\} \text{ and } (h, d) \text{ added by TR}
\]