“Training Deterministic Parsers with Non-Deterministic Oracles”

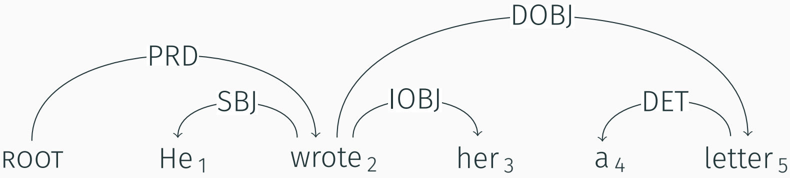
by Yoav Goldberg and Joakim Nivre, 2013

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



Transition System

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 $t : C \to C$,
3. c_s is an initialization function, mapping sentence $w = w_1 w_2 ... w_n$ 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

ROOT He $_1$ wrote $_2$ her $_3$ a $_4$ letter $_5$

$\top c_s(w)$

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

ROOT He $_1$ wrote $_2$ her $_3$ a $_4$ letter $_5$

[ROOT], [He$_1$, wrote$_2$, her$_3$, a$_4$, letter$_5$]

Shift

[ROOT, He$_1$], [wrote$_2$, her$_3$, a$_4$, letter$_5$]

SBJ

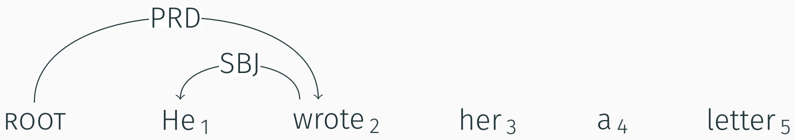
ROOT He $_1$ wrote $_2$ her $_3$ a $_4$ letter $_5$

[ROOT, He$_1$], [wrote$_2$, her$_3$, a$_4$, letter$_5$]

$\textsc{Left}_{\text{SBJ}}$

[ROOT], [wrote$_2$, her$_3$, a$_4$, letter$_5$]

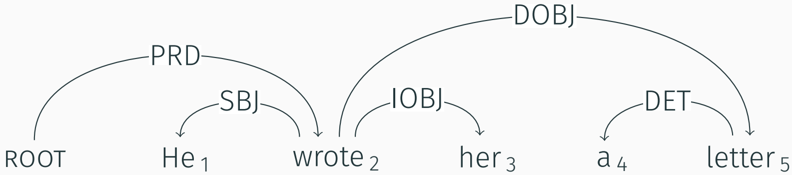
Training Deterministic Parsers with Non-Deterministic Oracles



[ROOT], [wrote_2, her_3, a_4, letter_5]

$\vdash$ $\text{RIGHT}_{\text{PRD}}$

[ROOT, wrote_2], [her_3, a_4, letter_5]



$[\textsc{root}, \text{wrote}_2], [\text{her}_3, \text{a}_4, \text{letter}_5]$

$\top$ $\textsc{Right}_{\text{IOBJ}}, \textsc{Shift}, \textsc{Left}_{\text{DET}}, \textsc{Reduce}, \textsc{Right}_{\text{DOBJ}}$

$[\textsc{root}, \text{wrote}_2, \text{letter}_5], [\,] \in C_t$

Training Deterministic Parsers with Non-Deterministic Oracles

if $c = (\sigma|i, j|\beta, A)$ *and* $(j, i) \in T$ **then**
    $t \leftarrow$ LEFT
else if $c = (\sigma|i, j|\beta, A)$ *and* $(i, j) \in T$ **then**
    $t \leftarrow$ RIGHT
else if $c = (\sigma|i, j|\beta, A)$ *and* $\exists k[k < i \wedge [(k, j) \in T \vee (j, k) \in T]]$ **then**
    $t \leftarrow$ REDUCE
else
    $t \leftarrow$ SHIFT
return t

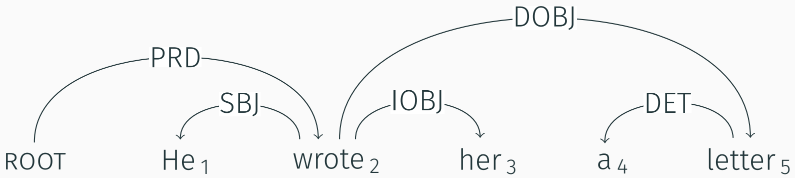
Greedy Classifier-based Parsing

$c \leftarrow c_s(w)$
while $c \notin C_t$ **do**
  $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
    $c \leftarrow t_p(c)$
return A_c

Training Deterministic Parsers with Non-Deterministic Oracles

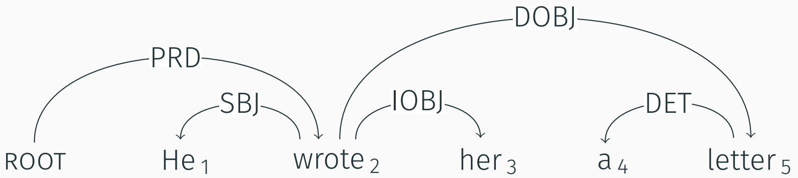
for $(w, T) \in d$ **do**
$\quad c \leftarrow c_s(w)$
$\quad$ **while** $c \notin C_t$ **do**
$\quad\quad t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
$\quad\quad \textsc{Correct}(c) \leftarrow \{t \mid o(t; c, T) = \textbf{true}\}$
$\quad\quad t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
$\quad\quad$ **if** $t_p \notin \textsc{Correct}(c)$ **then**
$\quad\quad\quad \textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
$\quad\quad\quad c \leftarrow t_o(c)$
$\quad\quad$ **else**
$\quad\quad\quad c \leftarrow t_p(c)$
return $\mathbf{w}$

Training Deterministic Parsers with Non-Deterministic Oracles



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

Training Deterministic Parsers with Non-Deterministic Oracles

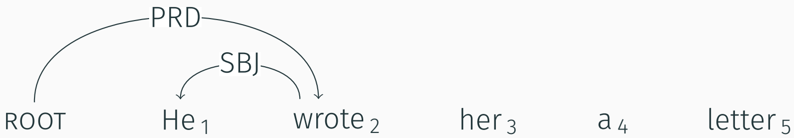


SH, LA_{SBJ}, RA_{PRD}, RA_{IOBJ}, SH, LA_{DET}, RE, RA_{DOBJ}

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

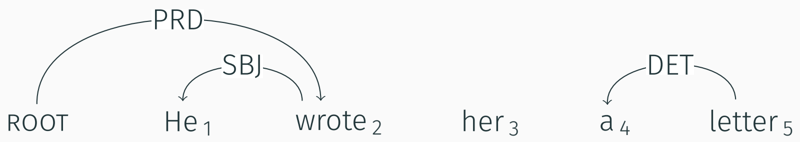


$[\text{ROOT}], [\text{He}_1, \text{wrote}_2, \text{her}_3, \text{a}_4, \text{letter}_5]$

$\vdash$ $\text{SH}, \text{LA}_{\text{SBJ}}, \text{RA}_{PRD}, \text{SH}$

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

… with Non-Deterministic and Complete Oracles



$$[\textsc{root}, \text{wrote}_2, \text{her}_3], [\text{a}_4, \text{letter}_5]$$

$$\vdash \text{SH}, \text{LA}_{\text{DET}}, \text{SH}$$

$$[\textsc{root}, \text{wrote}_2, \text{her}_3, \text{letter}_5], [\,] \in C_t$$

→ error propagation can be mitigated by complete oracle

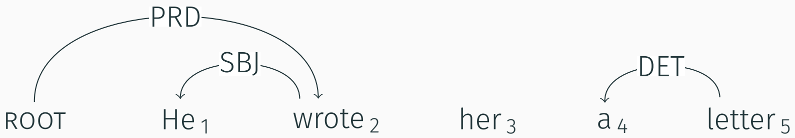
→ dynamic oracle: non-deterministic + complete

Training (Standard)

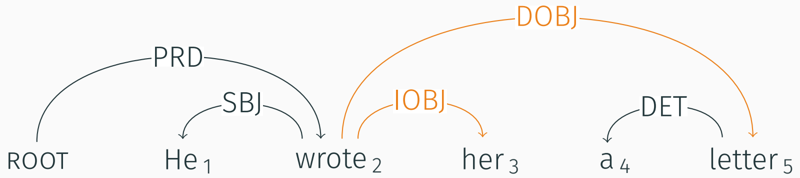
for $(w, T) \in d$ **do**
    $c \leftarrow c_s(w)$
    while $c \notin C_t$ **do**
        $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
        $\textsc{Correct}(c) \leftarrow \{t \mid o(t; c, T) = \textbf{true}\}$
        $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
        if $t_p \notin \textsc{Correct}(c)$ **then**
            $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
            $c \leftarrow t_o(c)$
        else
            $c \leftarrow t_p(c)$
return $\mathbf{w}$

Training with Exploration

for $(w, T) \in d$ **do**
    $c \leftarrow c_s(w)$
    while $c \notin C_t$ **do**
        $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
        $\textsc{Optimal}(c) \leftarrow \{t \mid o(t; c, T) = \mathbf{true}\}$
        $t_o \leftarrow \arg\max_{t \in \textsc{Optimal}(c)} \mathbf{w} \cdot \phi(c, t)$
        if $t_p \notin \textsc{Optimal}(c)$ **then**
            $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
            $c \leftarrow \textsc{Explore}(c, t_o, t_p)$
        else
            $c \leftarrow t_p(c)$
return $\mathbf{w}$

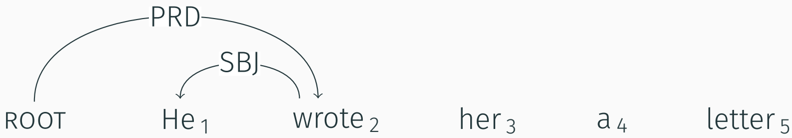

PRD
SBJ
DET
ROOT
He 1
wrote 2
her 3
a 4
letter 5

Optimality / Transition Costs



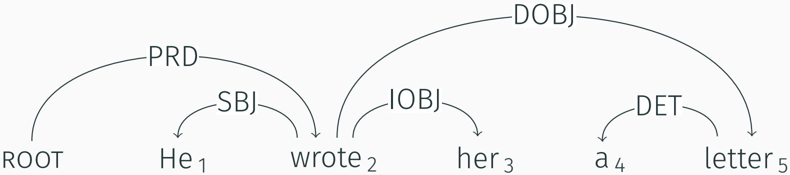
$$\mathcal{C}(A, T) = 2$$

Optimality / Transition Costs

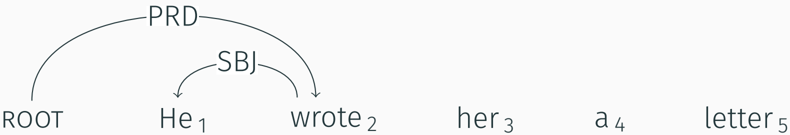


$[\textsc{root}, \text{wrote}_2, \text{her}_3], [\text{a}_4, \text{letter}_5]$

Optimality / Transition Costs



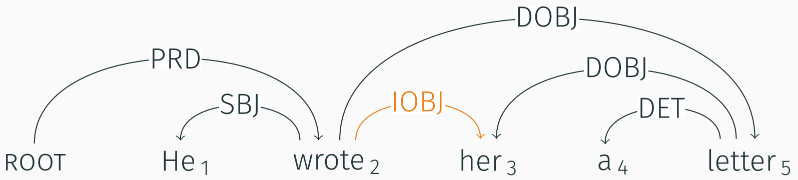
$$\min_{A: c \rightsquigarrow A} \mathcal{C}(A, T) = 0$$



[ROOT, wrote$_2$, her$_3$], [a$_4$, letter$_5$]

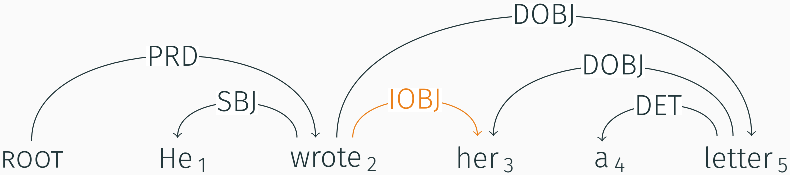
SH, ...

Optimality / Transition Costs



$$\mathcal{C}(\textsc{Shift}; c, T) = \min_{A:t(c)\rightsquigarrow A} \mathcal{C}(A, T) - \min_{A:c\rightsquigarrow A} \mathcal{C}(A, T) = 1$$

Optimality / Transition Costs



$$\mathcal{C}(\textsc{Shift}; c, T) = \min_{A:t(c)\rightsquigarrow A} \mathcal{C}(A, T) - \min_{A:c\rightsquigarrow A} \mathcal{C}(A, T) = 1$$

$$o_d(c, T) = \{t \mid \mathcal{C}(t; c, T) = 0\}$$

Arc Decomposition - Definition

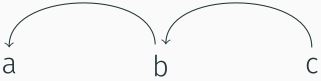
Definition (Tree Consistency)

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 $(h, d) \in A$.

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



Arc-Standard Transitions

$\textsc{Left}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})$

$\textsc{Right}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})$

$\textsc{Shift}[(\sigma, b|\beta, A)] = (\sigma|b, \beta, A)$

Arc Decomposition - Arc-Standard Counterexample

$$c = ([a, b, c], \beta) \overset{\textsc{Left}}{\vdash} ([a, c], \beta)$$

Arc-Standard Transitions

$$\textsc{Left}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})$$
$$\textsc{Right}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})$$
$$\textsc{Shift}[(\sigma, b|\beta, A)] = (\sigma|b, \beta, A)$$

Arc Decomposition - Arc-Standard Counterexample

$$c = ([a, b, c], \beta) \overset{\textsc{Right}}{\vdash} ([a, b], \beta) \overset{\textsc{Left}}{\vdash} ([b], \beta)$$

Arc-Standard Transitions

$\textsc{Left}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_0, \beta, A \cup \{(s_0, s_1)\})$

$\textsc{Right}[(\sigma|s_1|s_0, \beta, A)] = (\sigma|s_1, \beta, A \cup \{(s_1, s_0)\})$

$\textsc{Shift}[(\sigma, b|\beta, A)] = (\sigma|b, \beta, A)$

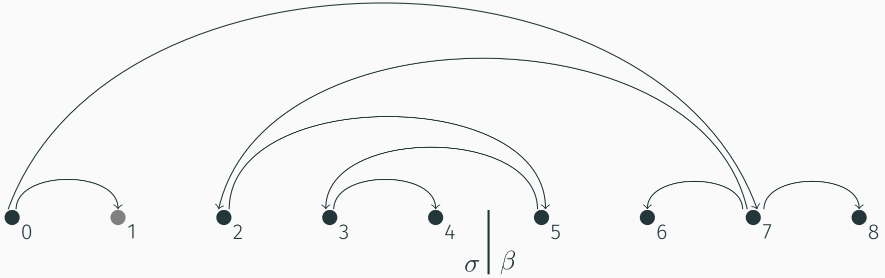
Arc Decomposition - Arc-Eager Proof Sketch

Given: arbitrary configuration $c = (\sigma, \beta, A)$ and tree consistent arc set A' such that all arc are reachable from c.

To show: $c \rightsquigarrow A'$

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

Arc Decomposition - Arc-Eager Proof Sketch



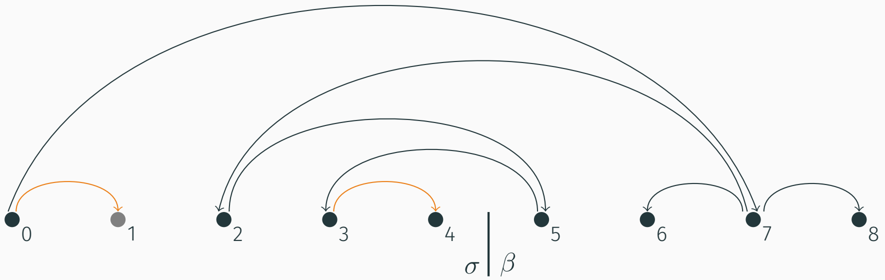
$$\overline{\mathcal{B}} = \{(h, d) \mid h, d \notin \beta\}$$

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

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

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

Arc Decomposition - Arc-Eager Proof Sketch

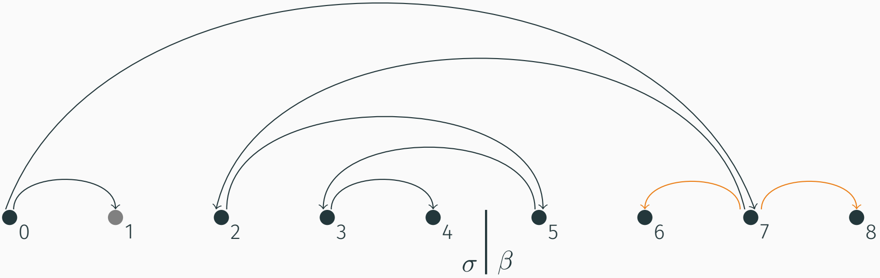


$$\overline{\mathcal{B}} = \{(h,d) \mid h,d \notin \beta\}$$

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

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

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

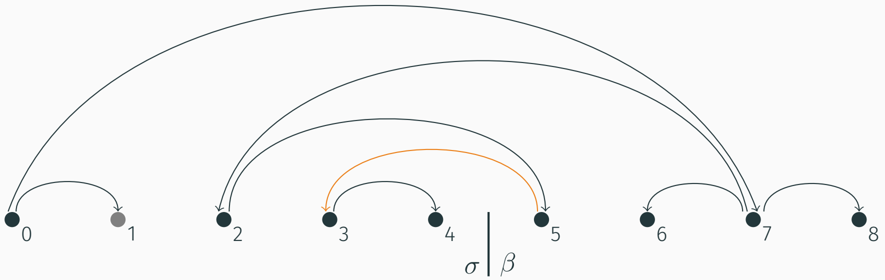


$$\overline{\mathcal{B}} = \{(h, d) \mid h, d \notin \beta\}$$

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

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

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



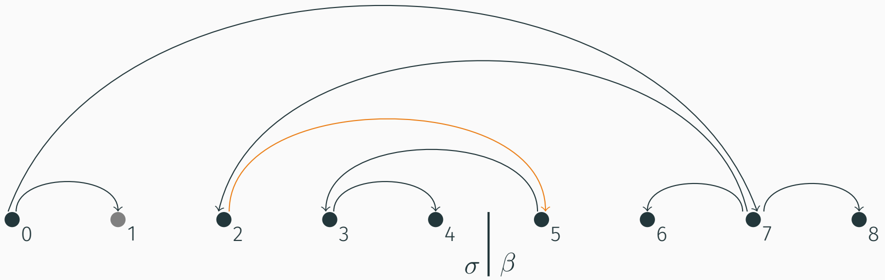
$$\overline{\mathcal{B}} = \{(h, d) \mid h, d \notin \beta\}$$

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

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

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

Arc Decomposition - Arc-Eager Proof Sketch



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

Dynamic Oracle

$$o_d(c, T) = \{t \mid \mathcal{C}(t; c, T) = 0\}$$

$$\mathcal{C}(t; c, T) = \min_{A:t(c) \rightsquigarrow A} \mathcal{C}(A, T) - \min_{A:c \rightsquigarrow A} \mathcal{C}(A, T)$$

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

Yoav Goldberg and Michael Elhadad. "An Efficient Algorithm for Easy-First Non-Directional Dependency Parsing". In: *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 2-4, 2010, Los Angeles, California, USA*. 2010, pp. 742–750. URL: http://www.aclweb.org/anthology/N10-1115.

Yoav Goldberg and Joakim Nivre. “A Dynamic Oracle for Arc-Eager Dependency Parsing”. In: *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*. 2012, pp. 959–976. URL: `http://aclweb.org/anthology/C/C12/C12-1059.pdf`.

Yoav Goldberg and Joakim Nivre. “Training Deterministic Parsers with Non-Deterministic Oracles”. In: *TACL* 1 (2013), pp. 403–414. URL: `https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/145`.

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. “A Tabular Method for Dynamic Oracles in Transition-Based Parsing”. In: *TACL* 2 (2014), pp. 119–130. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/302.

References iv

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. “Dynamic Programming Algorithms for Transition-Based Dependency Parsers”. In: *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*. 2011, pp. 673–682. URL: `http://www.aclweb.org/anthology/P11-1068`.

Joakim Nivre. “An Efficient Algorithm for Projective Dependency Parsing”. In: *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*. 2003, pp. 149–160.

Joakim Nivre. “Incrementality in Deterministic Dependency Parsing”. In: *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. 2004. URL: http://www.aclweb.org/anthology/W04-0308.

Joakim Nivre. “Algorithms for Deterministic Incremental Dependency Parsing”. In: *Computational Linguistics* 34.4 (2008), pp. 513–553. DOI: 10.1162/coli.07-056-R1-07-027. URL: https://doi.org/10.1162/coli.07-056-R1-07-027.

Transition Costs - Arc-Eager

$$
\begin{aligned}
\mathcal{C}(\textsc{Shift}; (\sigma, b|\beta, A), T) &= |\{(k, b) \in T \mid k \in \sigma\} \\
&\quad \cup \{(b, k) \in T \mid k \in \sigma \wedge \forall x \in V : (x, k) \notin A\}| \\
\mathcal{C}(\textsc{Right}; (\sigma|s, b|\beta, A), T) &= |\{(k, b) \in T \mid k \in \sigma \cup \beta\} \\
&\quad \cup \{(b, k) \in T \mid k \in \sigma \wedge \forall x \in V : (x, k) \notin A\}| \\
\mathcal{C}(\textsc{Left}; (\sigma|s, b|\beta, A), T) &= |\{(k, s) \in T \mid k \in \beta\} \cup \{(s, k) \in T \mid k \in \beta\}| \\
\mathcal{C}(\textsc{Reduce}; (\sigma|s, \beta, A), T) &= |\{(s, k) \in T \mid k \in \beta\}|
\end{aligned}
$$

Transition Costs - Hybrid

$$
\begin{aligned}
\mathcal{C}(\textsc{Shift}; (\sigma|s_1|s_0, b|\beta, A), T) &= |\{(b, k) \in T \mid k \in \{s_0, s_1\} \cup \sigma\} \\
&\quad \cup \{(k, b) \in T \mid k \in \{s_1\} \cup \sigma\}| \\
\mathcal{C}(\textsc{Right}; (\sigma|s_1|s_0, \beta, A), T) &= |\{(s_0, k) \in T \mid k \in \beta\} \\
&\quad \cup \{(k, s_0) \in T \mid k \in \beta\}| \\
\mathcal{C}(\textsc{Left}; (\sigma|s_1|s_0, b|\beta, A), T) &= |\{(s_0, k) \in T \mid k \in \{b\} \cup \beta\} \\
&\quad \cup \{(k, s_0) \in T \mid k \in \{s_1\} \cup \beta\}|
\end{aligned}
$$

Transition Costs - Easy-First

$$\mathcal{C}(\textsc{Tr}; (\lambda, A), T) = |\{(h', d) \in T \mid h' \in \lambda \wedge h' \neq h\} \cup \{(d, d') \in T \mid d' \in \lambda\}|$$

$$\textsc{Tr} \in \{\textsc{Left}^{i}_{lb} \mid 1 < i \leq |\lambda|\} \cup \{\textsc{Right}^{i}_{lb} \mid 1 \leq i < |\lambda|\} \text{ and } (h, d) \text{ added by } \textsc{Tr}$$