# Advanced Lecture on Dependency Parsing:
## Dynamic Oracles and Online Reordering

Miryam de Lhoneux

UPPSALA
UNIVERSITET

14 December 2017

# Outline

# Outline for section 1

# Transition-Based Parsing with Arc-Hybrid

Configuration:

| STACK | BUFFER |
| --- | --- |
| Drive your | friend home root |

Transitions:

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:

| STACK | BUFFER |
|---|---|
| Drive your | friend home root |

Transitions:

LEFT–ARC

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:

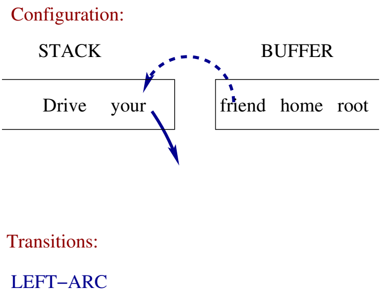
| STACK | BUFFER |
| --- | --- |
| Drive your | friend home root |

Transitions:

LEFT–ARC

Kuhlmann et al. (2011)

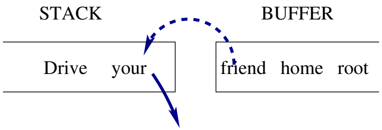
# Transition-Based Parsing with Arc-Hybrid




Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:




Transitions:

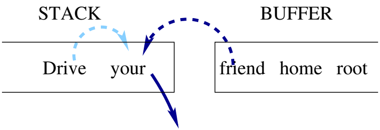
LEFT−ARC

RIGHT−ARC

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:




Transitions:

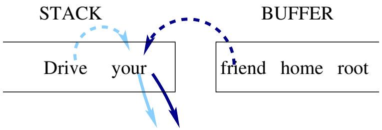
LEFT−ARC

RIGHT−ARC

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:




Transitions:

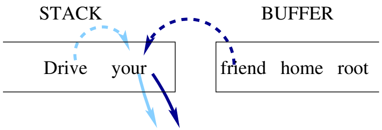
LEFT–ARC

RIGHT–ARC

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:




Transitions:

LEFT−ARC

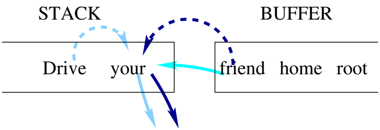
RIGHT−ARC

SHIFT

Kuhlmann et al. (2011)

# Transition-Based Parsing with Arc-Hybrid

Configuration:




Transitions:

LEFT−ARC

RIGHT−ARC

SHIFT

Kuhlmann et al. (2011)

# Training a Transition-Based Parser

**Algorithm 2** Online training with a static oracle

$\mathbf{w} \leftarrow 0$
**for** $I = 1 \rightarrow$ ITERATIONS **do**
  **for** sentence $x$ with gold tree $G_{\text{gold}}$ in corpus **do**
    $c \leftarrow c_s(x)$
    **while** $c$ is not terminal **do**
      $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$
      $t_o \leftarrow o(c, G_{\text{gold}})$
      **if** $t_p \neq t_o$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
      $c \leftarrow t_o(c)$
**return w**

**Figure :** Figure taken from Goldberg and Nivre (2012)

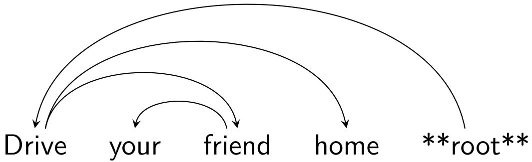
# Training a Transition-Based Parser

**Algorithm 2** Online training with a static oracle

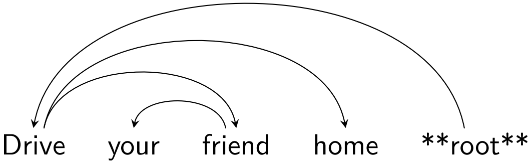
**w** $\leftarrow 0$
**for** $I = 1 \rightarrow$ ITERATIONS **do**
  **for** sentence $x$ with gold tree $G_{\text{gold}}$ in corpus **do**
    $c \leftarrow c_s(x)$
    **while** $c$ is not terminal **do**
      $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$
      $t_o \leftarrow \boxed{o(c, G_{\text{gold}})}$ oracle: a function that gives the correct transition for a configuration
      **if** $t_p \neq t_o$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
      $c \leftarrow t_o(c)$
**return w**

**Figure :** Figure taken from Goldberg and Nivre (2012)

# Static Oracle for Arc-Hybrid



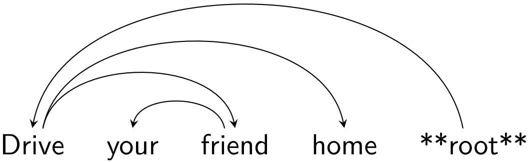
# Static Oracle for Arc-Hybrid




[ ]          [Drive your friend home **root**]

# Static Oracle for Arc-Hybrid

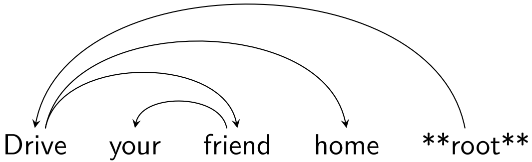
SHIFT




[ Drive ] [your friend home **root**]

# Static Oracle for Arc-Hybrid

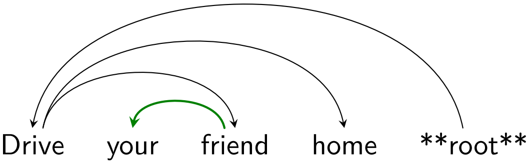
SHIFT




[ Drive your ] [friend home **root**]

# Static Oracle for Arc-Hybrid

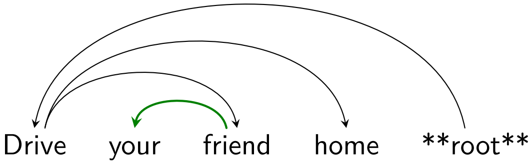
LEFT-ARC




[ Drive ] [friend home **root**]

# Static Oracle for Arc-Hybrid

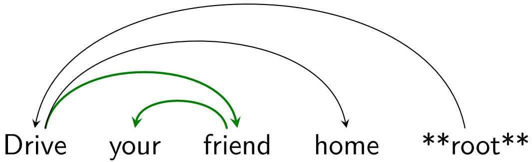
SHIFT




[ Drive friend]  [home **root**]

# Static Oracle for Arc-Hybrid

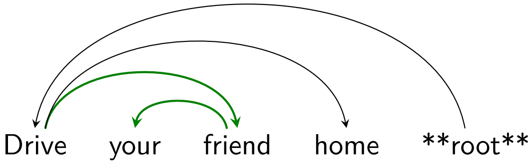
RIGHT-ARC




[ Drive ]          [home **root**]

# Static Oracle for Arc-Hybrid

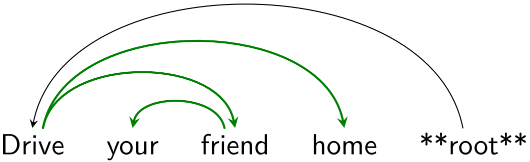
SHIFT




[ Drive home] [**root**]

# Static Oracle for Arc-Hybrid

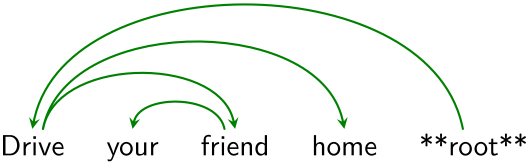
RIGHT-ARC




[ Drive ]          [**root**]

# Static Oracle for Arc-Hybrid

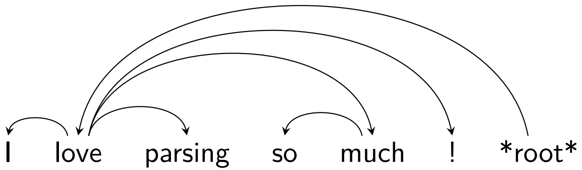
LEFT-ARC




[ ] [**root**]

# Exercise!

Write the transition sequence for this tree:



# Outline for section 2

# Static Oracle: problem 1

**Algorithm 2** Online training with a static oracle

$\mathbf{w} \leftarrow 0$
**for** $I = 1 \rightarrow$ ITERATIONS **do**
  **for** sentence $x$ with gold tree $G_{\text{gold}}$ in corpus **do**
    $c \leftarrow c_s(x)$
    **while** $c$ is not terminal **do**
      $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$
      $t_o \leftarrow o(c, G_{\text{gold}})$
      **if** $t_p \neq t_o$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
      $c \leftarrow t_o(c)$
**return w**

**Figure :** Figure taken from Goldberg and Nivre (2012)

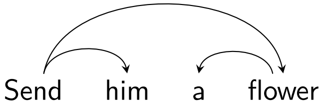
# Static Oracle: problem 1

**Algorithm 2** Online training with a static oracle

```
w ← 0
for I = 1 → ITERATIONS do
    for sentence x with gold tree G_gold in corpus do
        c ← c_s(x)
        while c is not terminal do
            t_p ← argmax_t w · φ(c, t)
            [t_o ← o(c, G_gold)]        consider one correct transition
            if t_p ≠ t_o then
                w ← w + φ(c, t_o) − φ(c, t_p)
            c ← t_o(c)
return w
```

**Figure :** Figure taken from Goldberg and Nivre (2012)

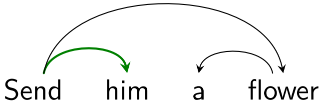
# Spurious ambiguity




[ Send him ] [ a flower ]

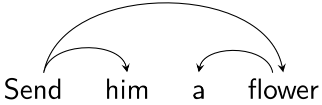
# Spurious ambiguity

RIGHT-ARC




[ Send ] [ a flower ]

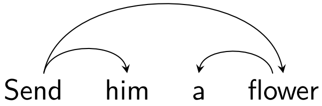
# Spurious ambiguity




[ Send him ] [ a flower ]

# Spurious ambiguity

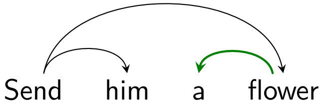
SHIFT




[ Send him a ]    [ flower ]

# Spurious ambiguity

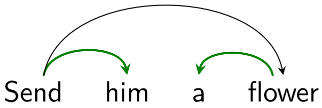
LEFT-ARC




[ Send him ] [ flower ]

# Spurious ambiguity

RIGHT-ARC




[ Send ] [ flower ]

# Static Oracle: problem 2

**Algorithm 2** Online training with a static oracle

$\mathbf{w} \leftarrow 0$<br>
**for** $I = 1 \rightarrow$ ITERATIONS **do**<br>
    **for** sentence $x$ with gold tree $G_{\text{gold}}$ in corpus **do**<br>
        $c \leftarrow c_s(x)$<br>
        **while** $c$ is not terminal **do**<br>
            $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$<br>
            $t_o \leftarrow o(c, G_{\text{gold}})$<br>
            **if** $t_p \neq t_o$ **then**<br>
                $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$<br>
            $c \leftarrow t_o(c)$<br>
**return w**

**Figure :** Figure taken from Goldberg and Nivre (2012)

# Static Oracle: problem 2

**Algorithm 2** Online training with a static oracle

**$\mathbf{w} \leftarrow 0$**
**for** $I = 1 \rightarrow$ ITERATIONS **do**
  **for** sentence $x$ with gold tree $G_{\text{gold}}$ in corpus **do**
    $c \leftarrow c_s(x)$
    **while** $c$ is not terminal **do**
      $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c,t)$
      $t_o \leftarrow o(c, G_{\text{gold}})$
      **if** $t_p \neq t_o$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + \phi(c,t_o) - \phi(c,t_p)$
      $\boxed{c \leftarrow t_o(c)}$ We always apply the correct transition
We only see 'gold' configurations in training!
**return w**

**Figure :** Figure taken from Goldberg and Nivre (2012)

# Training with a Dynamic Oracle

**Algorithm 3** Online training with exploration for greedy transition-based parsers ($i$th iteration)

**for** sentence $W$ with gold tree $T$ in corpus **do**
  $c \leftarrow \textsc{Initial}(W)$
  **while not** $\textsc{Terminal}(c)$ **do**
    $\textsc{Correct}(c) \leftarrow \{t | o(t; c, T) = \textbf{true}\}$
    $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
    $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
    **if** $t_p \notin \textsc{Correct}(c)$ **then**
      $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
      $c \leftarrow \textsc{Explore}_{k,p}(c, t_o, t_p, i)$
    **else**
      $c \leftarrow t_p(c)$

**function** $\textsc{Explore}_{k,p}(c, t_o, t_p, i)$
  **if** $i > k$ **and** $\textsc{Rand}() < p$ **then**
    **return** $t_p(c)$
  **else**
    **return** $\textsc{Next}(c, t_o)$

**Figure :** Figure taken from Goldberg and Nivre (2013)

# Training with a Dynamic Oracle

**Algorithm 3** Online training with exploration for greedy transition-based parsers ($i$th iteration)

**for** sentence $W$ with gold tree $T$ in corpus **do**
  $c \leftarrow \textsc{Initial}(W)$
  **while not** $\textsc{Terminal}(c)$ **do**
    $\boxed{\textsc{Correct}(c) \leftarrow \{t | o(t; c, T) = \textbf{true}\}}$
    $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
    $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
    **if** $t_p \notin \textsc{Correct}(c)$ **then**
      $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
      $c \leftarrow \textsc{Explore}_{k,p}(c, t_o, t_p, i)$
    **else**
      $c \leftarrow t_p(c)$

We consider all correct transitions

**function** $\textsc{Explore}_{k,p}(c, t_o, t_p, i)$
  **if** $i > k$ **and** $\textsc{Rand}() < p$ **then**
    **return** $t_p(c)$
  **else**
    **return** $\textsc{Next}(c, t_o)$

**Figure :** Figure taken from Goldberg and Nivre (2013)

# Training with a Dynamic Oracle

**Algorithm 3** Online training with exploration for greedy transition-based parsers ($i$th iteration)

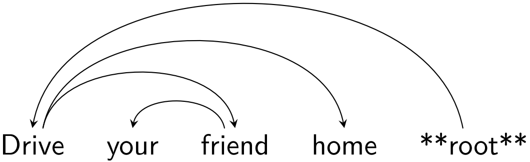
```
for sentence W with gold tree T in corpus do
    c ← INITIAL(W)
    while not TERMINAL(c) do
        CORRECT(c) ← {t | o(t; c, T) = true}
        t_p ← arg max_{t∈LEGAL(c)} w · φ(c, t)
        t_o ← arg max_{t∈CORRECT(c)} w · φ(c, t)
        if t_p ∉ CORRECT(c) then
            UPDATE(w, φ(c, t_o), φ(c, t_p))
            c ← [EXPLORE_{k,p}(c, t_o, t_p, i)]
        else
            c ← t_p(c)
```

We allow taking incorrect transitions.
The oracle must be defined over 'erroneous' configurations.

```
function EXPLORE_{k,p}(c, t_o, t_p, i)
    if i > k and RAND() < p then
        return t_p(c)
    else
        return NEXT(c, t_o)
```

**Figure :** Figure taken from Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

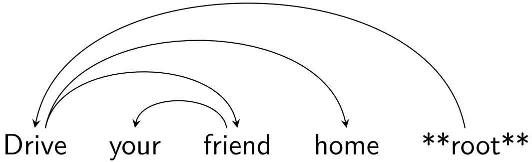



[ Drive your ]        [ friend home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

RIGHT-ARC

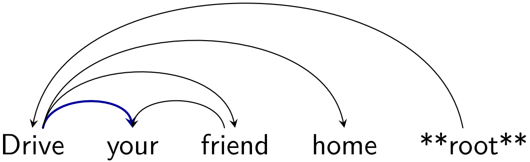



[ Drive your ] [ friend home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

RIGHT-ARC

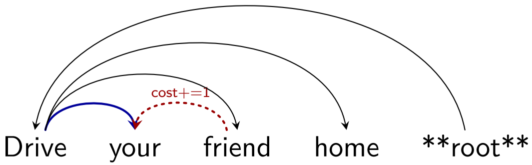



[ Drive ]    [ friend home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

RIGHT-ARC

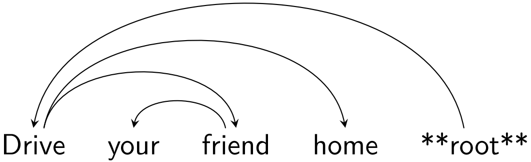



[ Drive ] [ friend home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

SHIFT

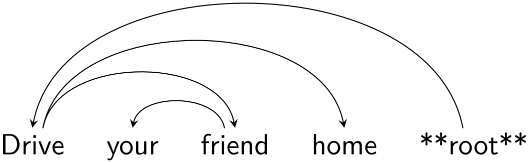



[ Drive your ] [ friend home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

SHIFT

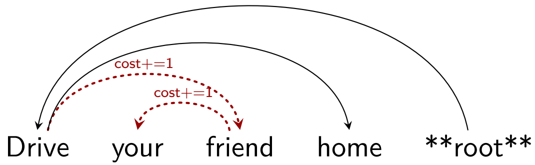



[ Drive your friend ]        [ home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

SHIFT




[ Drive your friend ] [ home **root**]

Goldberg and Nivre (2013)

# Dynamic Oracle for Arc-Hybrid

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc $(b, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads from $H = \{s_1\} \cup \beta$ and will not be able to acquire dependents from $D = \{b\} \cup \beta$. The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h \in H$ and $d \in D$.

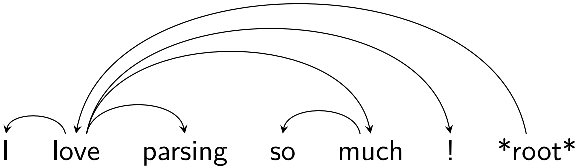
- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc $(s_1, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads or dependents from $B = \{b\} \cup \beta$. The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h, d \in B$.

- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing $b$ onto the stack means that $b$ will not be able to acquire heads from $H = \{s_1\} \cup \sigma$, and will not be able to acquire dependents from $D = \{s_0, s_1\} \cup \sigma$. The cost is therefore the number of arcs in $T$ of the form $(b, d)$ and $(h, b)$ for $h \in H$ and $d \in D$.

**Figure :** Cost function

# Exercise!

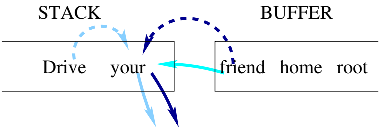
For the 7 first transitions in the correct transition sequence we defined in the previous exercise, compute the cost of the other legal transitions.



# Outline for section 3

# Arc-Hybrid Parsing with Reordering

Configuration:




Transitions:

LEFT−ARC

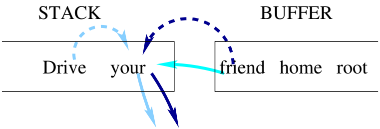
RIGHT−ARC

SHIFT

Nivre (2009); de Lhoneux et al. (2017)

# Arc-Hybrid Parsing with Reordering

Configuration:




Transitions:

LEFT−ARC

RIGHT−ARC

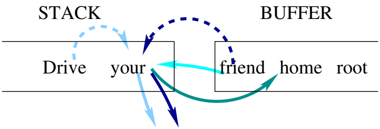
SHIFT

SWAP

Nivre (2009); de Lhoneux et al. (2017)

# Arc-Hybrid Parsing with Reordering

Configuration:




Transitions:

LEFT−ARC

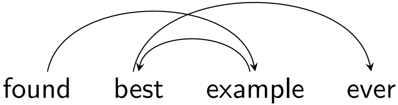
RIGHT−ARC

SHIFT

SWAP

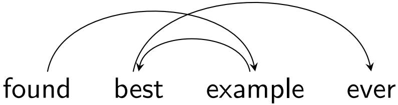
Nivre (2009); de Lhoneux et al. (2017)

# Arc-Hyrid Parsing with Reordering

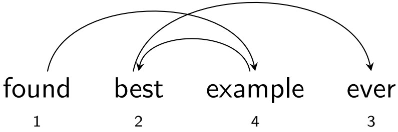



(Thanks Carlos Gomez-Rodriguez for the example!)

# Arc-Hyrid Parsing with Reordering

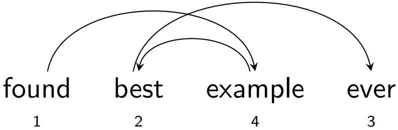


# Arc-Hyrid Parsing with Reordering



# Arc-Hyrid Parsing with Reordering

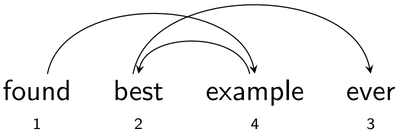
SHIFT




[ ] [ found$_1$ best$_2$ example$_4$ ever$_3$ ]

# Arc-Hyrid Parsing with Reordering

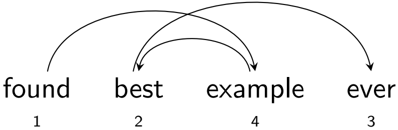
SHIFT




[ found$_1$ ]  [ best$_2$ example$_4$ ever$_3$ ]

# Arc-Hyrid Parsing with Reordering

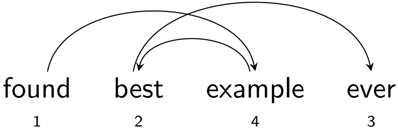
SHIFT




[ found$_1$ best$_2$ ] [ example$_4$ ever$_3$ ]

# Arc-Hyrid Parsing with Reordering

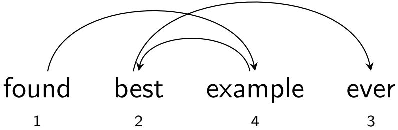
SHIFT




[ found$_1$ best$_2$ example$_4$ ]  [ ever$_3$ ]

# Arc-Hyrid Parsing with Reordering

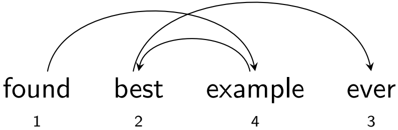
SHIFT




[ found$_1$ best$_2$ example$_4$ ]     [ ever$_3$ ]

# Arc-Hyrid Parsing with Reordering

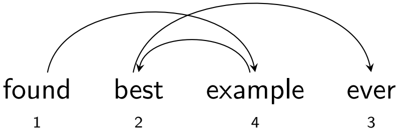
SWAP




[ found$_1$ best$_2$ ] [ ever$_3$ example$_4$ ]

# Arc-Hyrid Parsing with Reordering

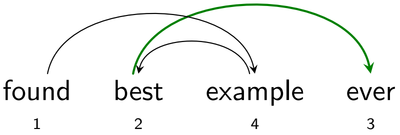
SHIFT




[ found$_1$ best$_2$ ever$_3$ ] [ example$_4$ ]

# Arc-Hyrid Parsing with Reordering

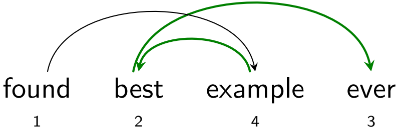
RIGHT-ARC




[ found$_1$ best$_2$ ]        [ example$_4$ ]

# Arc-Hyrid Parsing with Reordering

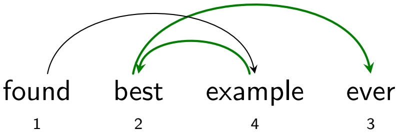
LEFT-ARC




[ found$_1$ ]  [ example$_4$ ]

# Arc-Hyrid Parsing with Reordering

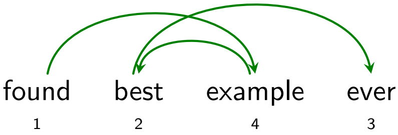
SHIFT




[ found$_1$ example$_4$ ]   [ ]

# Arc-Hyrid Parsing with Reordering

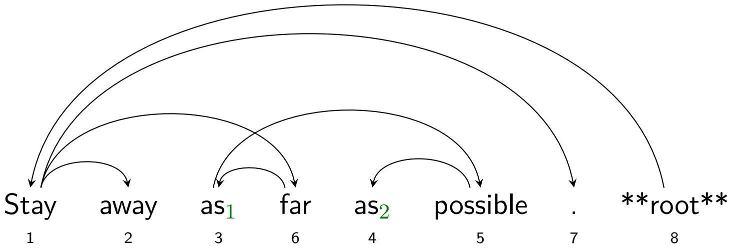
RIGHT-ARC




[ found$_1$ ]      [ ]

# Exercise!

Write the transition sequence for this tree:




(Example taken from the UD English treebank.)

# Outline for section 4

# Dynamic Oracles and Reordering

Dynamic oracle for parsing with reordering:
Open research question!

Partial solution:
A static-dynamic oracle (de Lhoneux et al., 2017)

# Dynamic Oracles and Reordering

Dynamic oracle for parsing with reordering:
Open research question!

Partial solution:
A static-dynamic oracle (de Lhoneux et al., 2017)

# Dynamic Oracles and Reordering

Dynamic oracle for parsing with reordering:
Open research question!

Partial solution:
A static-dynamic oracle (de Lhoneux et al., 2017)

# Dynamic Oracles and Reordering

Dynamic oracle for parsing with reordering:
Open research question!

Partial solution:
A static-dynamic oracle (de Lhoneux et al., 2017)

# Dynamic Oracles and Reordering

Dynamic oracle for parsing with reordering:
Open research question!

Partial solution:
A static-dynamic oracle (de Lhoneux et al., 2017)

# Why is it hard?

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc $(b, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads from $H = \{s_1\} \cup \beta$ and will not be able to acquire dependents from $D = \{b\} \cup \beta$. The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h \in H$ and $d \in D$.

- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc $(s_1, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads or dependents from $B = \{b\} \cup \beta$. The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h, d \in B$.

- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing $b$ onto the stack means that $b$ will not be able to acquire heads from $H = \{s_1\} \cup \sigma$, and will not be able to acquire dependents from $D = \{s_0, s_1\} \cup \sigma$. The cost is therefore the number of arcs in $T$ of the form $(b, d)$ and $(h, b)$ for $h \in H$ and $d \in D$.

# Why is it hard?

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc $(b, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \beta}$ and will not be able to acquire dependents from $\boxed{D = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h \in H$ and $d \in D$.

- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc $(s_1, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads or dependents from $\boxed{B = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h, d \in B$.

- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing $b$ onto the stack means that $b$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \sigma,}$ and will not be able to acquire dependents from $\boxed{D = \{s_0, s_1\} \cup \sigma.}$ The cost is therefore the number of arcs in $T$ of the form $(b, d)$ and $(h, b)$ for $h \in H$ and $d \in D$.

# Why is it hard?

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc $(b, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \beta}$ and will not be able to acquire dependents from $\boxed{D = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h \in H$ and $d \in D$.

- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc $(s_1, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads or dependents from $\boxed{B = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h, d \in B$.

- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing $b$ onto the stack means that $b$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \sigma,}$ and will not be able to acquire dependents from $\boxed{D = \{s_0, s_1\} \cup \sigma.}$ The cost is therefore the number of arcs in $T$ of the form $(b, d)$ and $(h, b)$ for $h \in H$ and $d \in D$.

Using information about the position of words in stack and buffer

# Why is it hard?

- $\mathcal{C}(\text{LEFT}; c, T)$: Adding the arc $(b, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \beta}$ and will not be able to acquire dependents from $\boxed{D = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h \in H$ and $d \in D$.

- $\mathcal{C}(\text{RIGHT}; c, T)$: Adding the arc $(s_1, s_0)$ and popping $s_0$ from the stack means that $s_0$ will not be able to acquire heads or dependents from $\boxed{B = \{b\} \cup \beta.}$ The cost is therefore the number of arcs in $T$ of the form $(s_0, d)$ and $(h, s_0)$ for $h, d \in B$.

- $\mathcal{C}(\text{SHIFT}; c, T)$: Pushing $b$ onto the stack means that $b$ will not be able to acquire heads from $\boxed{H = \{s_1\} \cup \sigma,}$ and will not be able to acquire dependents from $\boxed{D = \{s_0, s_1\} \cup \sigma.}$ The cost is therefore the number of arcs in $T$ of the form $(b, d)$ and $(h, b)$ for $h \in H$ and $d \in D$.

Using information about the position of words in stack and buffer
But now words can move!

# A Static-Dynamic Oracle

**Algorithm 3** Online training with exploration for greedy transition-based parsers ($i$th iteration)

**for** sentence $W$ with gold tree $T$ in corpus **do**
    $c \leftarrow \textsc{Initial}(W)$
    **while not** $\textsc{Terminal}(c)$ **do**
        $\textsc{Correct}(c) \leftarrow \{t | o(t; c, T) = \textbf{true}\}$
        $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
        $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
        **if** $t_p \notin \textsc{Correct}(c)$ **then**
            $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
            $c \leftarrow \textsc{Explore}_{k,p}(c, t_o, t_p, i)$
        **else**
            $c \leftarrow t_p(c)$

**function** $\textsc{Explore}_{k,p}(c, t_o, t_p, i)$
    **if** $i > k$ **and** $\textsc{Rand}() < p$ **then**
        **return** $t_p(c)$
    **else**
        **return** $\textsc{Next}(c, t_o)$

**Figure :** Figure taken from Goldberg and Nivre (2013)

# A Static-Dynamic Oracle

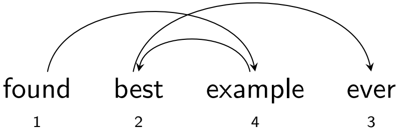
**Algorithm 3** Online training with exploration for greedy transition-based parsers ($i$th iteration)

**for** sentence $W$ with gold tree $T$ in corpus **do**
    $c \leftarrow \textsc{Initial}(W)$
    **while not** $\textsc{Terminal}(c)$ **do**
        $\textsc{Correct}(c) \leftarrow \{t | o(t; c, T) = \textbf{true}\}$
        $t_p \leftarrow \arg\max_{t \in \textsc{Legal}(c)} \mathbf{w} \cdot \phi(c, t)$
        $t_o \leftarrow \arg\max_{t \in \textsc{Correct}(c)} \mathbf{w} \cdot \phi(c, t)$
        **if** $t_p \notin \textsc{Correct}(c)$ **then**
            $\textsc{Update}(\mathbf{w}, \phi(c, t_o), \phi(c, t_p))$
            $\boxed{c \leftarrow \textsc{Explore}_{k,p}(c, t_o, t_p, i)}$
        **else** We disallow this if the correct transition is swap
            $c \leftarrow t_p(c)$

**function** $\textsc{Explore}_{k,p}(c, t_o, t_p, i)$
    **if** $i > k$ **and** $\textsc{Rand}() < p$ **then**
        **return** $t_p(c)$
    **else**
        **return** $\textsc{Next}(c, t_o)$

**Figure :** Figure taken from Goldberg and Nivre (2013)

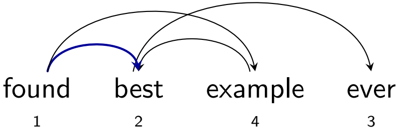
# A Static-Dynamic Oracle




[ found$_1$ best$_2$]　　　[ example$_4$ ever$_3$ ]

# A Static-Dynamic Oracle

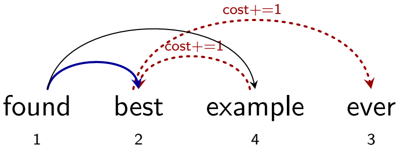
RIGHT-ARC




[ found$_1$ ]     [ example$_4$ ever$_3$ ]

# A Static-Dynamic Oracle

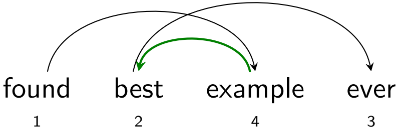
RIGHT-ARC




[ found$_1$ ]  [ example$_4$ ever$_3$ ]

# A Static-Dynamic Oracle

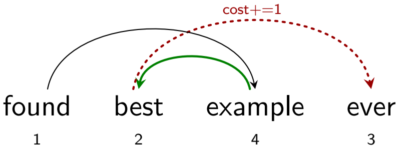
LEFT-ARC




[ found$_1$ ]     [ example$_4$ ever$_3$ ]

# A Static-Dynamic Oracle

LEFT-ARC




[ found$_1$ ]  [ example$_4$ ever$_3$ ]

# References


Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2017. Arc-hybrid non-projective dependency parsing with a static-dynamic oracle. In *Proceedings of the 15th International Conference on Parsing Technologies*. Association for Computational Linguistics, Pisa, Italy, pages 99–104. http://www.aclweb.org/anthology/W17-6314.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*. pages 959–976.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics* 1:403–414.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*. pages 673–682.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*. pages 351–359.