From Dependency Parsing to Imitation Learning

CMSC 723 / LING 723 / INST 725

Marine Carpuat

Fig credits: Joakim Nivre, Yoav Goldberg, Hal Daume III

Today's topics:
Addressing compounding error

- Improving on gold parse oracle
  - Research highlight: [Goldberg & Nivre, 2012]

- Imitation learning for structured prediction
  - CIML ch 18

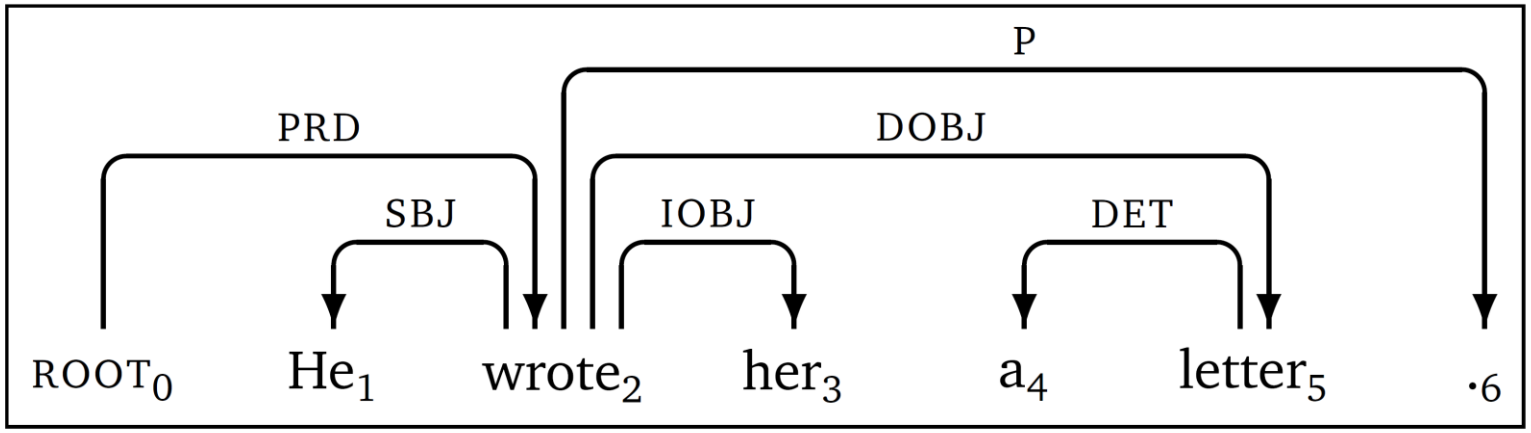
Improving the oracle in transition-based dependency parsing

- Issues with oracle we've used so far
  - Based on configuration sequence that produces gold tree
  - What if there are multiple sequences for a single gold tree?
  - How can we recover if the parser deviates from gold sequence?

- Goldberg & Nivre [2012] propose an improved oracle

A Dynamic Oracle for Arc-Eager Dependency Parsing

Yoav Goldberg[1] *Joakim Nivre*[1,2]
(1) Google Inc.
(2) Uppsala University
yogo@google.com, joakim.nivre@lingfil.uu.se



$$\text{SH, LA}_{\text{SBJ}}\text{, RA}_{\text{PRD}}\text{, RA}_{\text{IOBJ}}\text{, SH, LA}_{\text{DET}}\text{, RE, RA}_{\text{DOBJ}}\text{, RE RA}_{\text{P}}$$

$$\text{SH, LA}_{\text{SBJ}}\text{, RA}_{\text{PRD}}\text{, RA}_{\text{IOBJ}}\text{, RE, SH, LA}_{\text{DET}}\text{, RA}_{\text{DOBJ}}\text{, RE RA}_{\text{P}}$$

Exercise: which of these transition sequences produces the gold tree on the left?

Stack

Buffer

Dependency Arcs

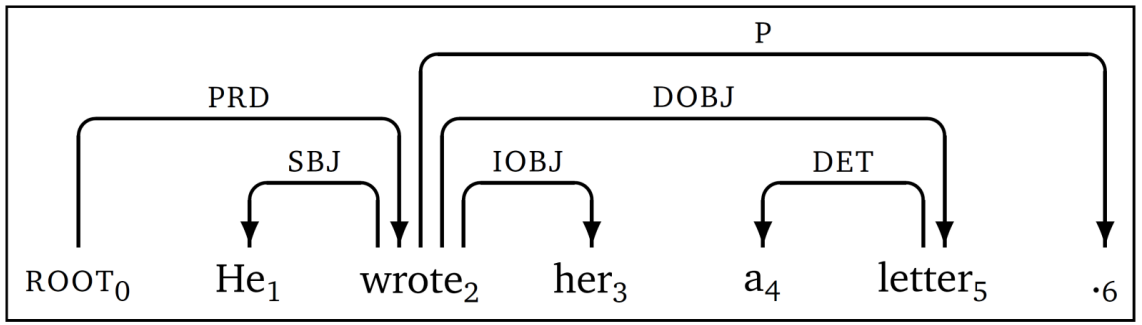
Arc from position j to position i, with dependency label l

Algorithm 1 Standard oracle for arc-eager dependency parsing

if $c = (\sigma|i, j|\beta, A)$ and $(j, l, i) \in A_{\text{gold}}$ **then**
    $t \leftarrow \text{LEFT-ARC}_l$
else if $c = (\sigma|i, j|\beta, A)$ and $(i, l, j) \in A_{\text{gold}}$ **then**
    $t \leftarrow \text{RIGHT-ARC}_l$
else if $c = (\sigma|i, j|\beta, A)$ and $\exists k[k < i \wedge \exists l[(k, l, j) \in A_{\text{gold}} \vee (j, l, k) \in A_{\text{gold}}]]$ **then**
    $t \leftarrow \text{REDUCE}$
else
    $t \leftarrow \text{SHIFT}$
return t

Algorithm 1 Standard oracle for arc-eager dependency parsing

1: **if** $c = (\sigma|i, j|\beta, A)$ and $(j, l, i) \in A_{\text{gold}}$ **then**
2: $\quad t \leftarrow \text{LEFT-ARC}_l$
3: **else if** $c = (\sigma|i, j|\beta, A)$ and $(i, l, j) \in A_{\text{gold}}$ **then**
4: $\quad t \leftarrow \text{RIGHT-ARC}_l$
5: **else if** $c = (\sigma|i, j|\beta, A)$ and $\exists k[k < i \wedge \exists l[(k, l, j) \in A_{\text{gold}} \vee (j, l, k) \in A_{\text{gold}}]]$ **then**
6: $\quad t \leftarrow \text{REDUCE}$
7: **else**
8: $\quad t \leftarrow \text{SHIFT}$
9: **return** t



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

Which of these transition sequences does the oracle algorithm produce?

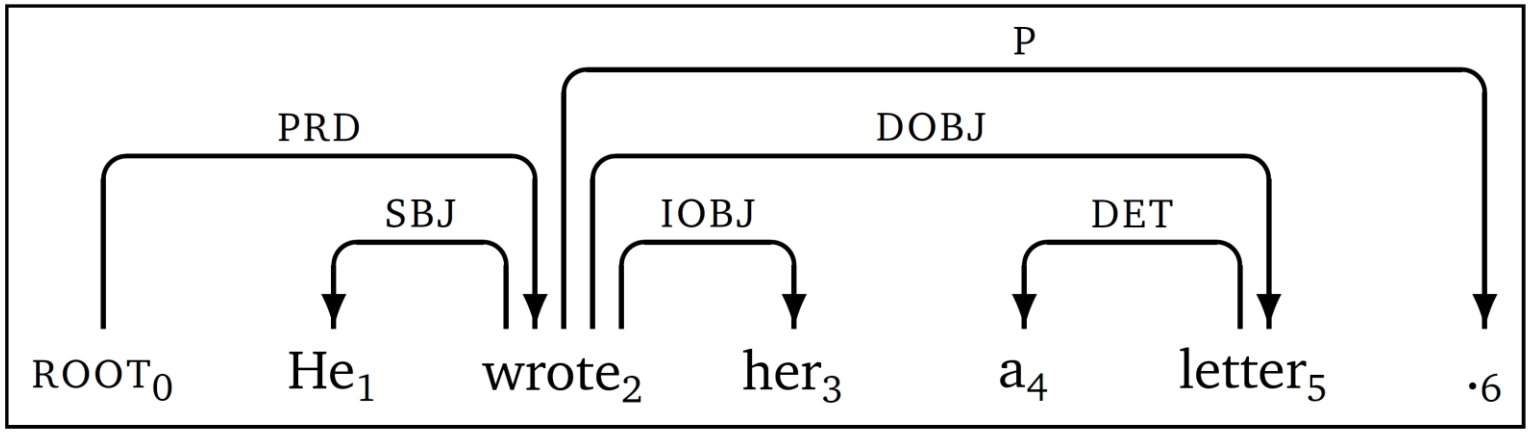
Improving the oracle in transition-based dependency parsing

- Issues with oracle we've used so far
  - Based on configuration sequence that produces gold tree
  - **What if there are multiple sequences for a single gold tree?**
  - How can we recover if the parser deviates from gold sequence?

- Goldberg & Nivre [2012] propose an improved oracle

A Dynamic Oracle for Arc-Eager Dependency Parsing

Yoav Goldberg[1] *Joakim Nivre*[1,2]
(1) Google Inc.
(2) Uppsala University
yogo@google.com, joakim.nivre@lingfil.uu.se

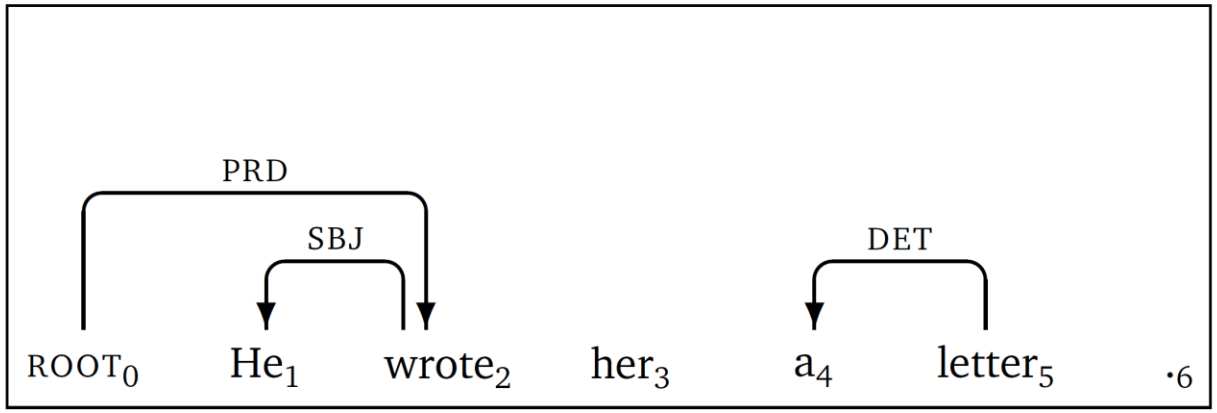


$\text{SH}, \text{LA}_{\text{SBJ}}, \text{RA}_{\text{PRD}}$, SHIFT

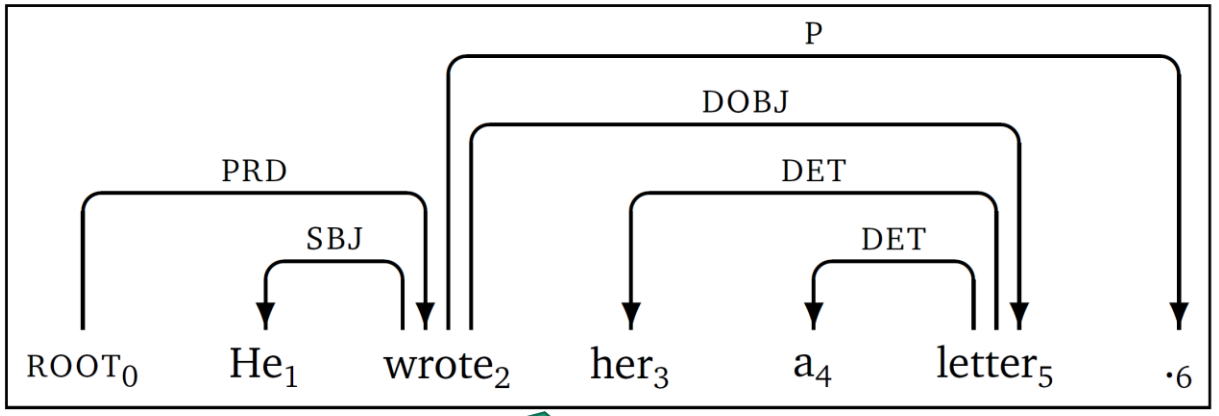
At test time, suppose the 4th transition predicted is SHIFT instead of RA_{IOBJ}
What happens if we apply the oracle next?

Measuring distance from gold tree

- **Labeled attachment loss:** number of arcs in gold tree that are not found in the predicted tree



Loss = 3



Loss = 1

Improving the oracle in transition-based dependency parsing

- Issues with oracle we've used so far
  - Based on configuration sequence that produces gold tree
  - What if there are multiple sequences for a single gold tree?
  - **How can we recover if the parser deviates from gold sequence?**

- Goldberg & Nivre [2012] propose an improved oracle

A Dynamic Oracle for Arc-Eager Dependency Parsing

Yoav Goldberg[1] *Joakim Nivre*[1,2]
(1) Google Inc.
(2) Uppsala University
yogo@google.com, joakim.nivre@lingfil.uu.se

Proposed solution:
2 key changes to training algorithm

Algorithm 3 Online training with a dynamic oracle

1: $\mathbf{w} \leftarrow 0$
2: **for** $I = 1 \rightarrow$ ITERATIONS **do**
3: **for** sentence x with gold tree G_{gold} in corpus **do**
4: $c \leftarrow c_s(x)$
5: **while** c is not terminal **do**
6: $t_p \leftarrow \arg\max_t \mathbf{w} \cdot \phi(c, t)$
7: ZERO_COST $\leftarrow \{t | o(t; c, G_{\text{gold}}) = \textbf{true}\}$
8: $t_o \leftarrow \arg\max_{t \in \text{ZERO\_COST}} \mathbf{w} \cdot \phi(c, t)$
9: **if** $t_p \notin$ ZERO_COST **then**
10: $\mathbf{w} \leftarrow \mathbf{w} + \phi(c, t_o) - \phi(c, t_p)$
11: $t_n \leftarrow$ CHOOSE_NEXT$(I, t_p,$ZERO_COST$)$
12: $c \leftarrow t_n(c)$
13: **return w**

Any transition that can possibly lead to a correct tree is considered correct

Explore non-optimal transitions

Proposed solution:
2 key changes to training algorithm

Algorithm 3 Online training with a dynamic oracle

```
1:  w ← 0
2:  for I = 1 → ITERATIONS do
3:      for sentence x with gold tree G_gold in corpus do
4:          c ← c_s(x)
5:          while c is not terminal do
6:              t_p ← arg max_t w · φ(c, t)
7:              ZERO_COST ← {t | o(t; c, G_gold) = true}
8:              t_o ← arg max_{t∈ZERO_COST} w · φ(c, t)
9:              if t_p ∉ ZERO_COST then
10:                 w ← w + φ(c, t_o) − φ(c, t_p)
11:             t_n ← CHOOSE_NEXT(I, t_p, ZERO_COST)
12:             c ← t_n(c)
13: return w
```

```
1: function CHOOSE_NEXT_AMB(I, t, ZERO_COST)
2:     if t ∈ ZERO_COST then
3:         return t
4:     else
5:         return RANDOM_ELEMENT(ZERO_COST)
```

```
1: function CHOOSE_NEXT_EXP(I, t, ZERO_COST)
2:     if I > k and RAND() > p then
3:         return t
4:     else
5:         return CHOOSE_NEXT_AMB(I, t, ZERO_COST)
```

Defining the cost of a transition

- Loss difference between minimum loss trees achievable before and after transition

$$\mathscr{C}(t; c, G_{\text{gold}}) = \left[\min_{G: t(c) \rightsquigarrow G} \mathscr{L}(G, G_{\text{gold}})\right] - \left[\min_{G: c \rightsquigarrow G} \mathscr{L}(G, G_{\text{gold}})\right]$$

- Loss for trees nicely decomposes into losses for arcs
  - We can compute transition cost by counting gold arcs that are no longer reachable after transition

Today's topics
Addressing compounding error

- Improving on gold parse oracle
  - Research highlight: [Goldberg & Nivre, 2012]

- Imitation learning for structured prediction
  - CIML ch 18

Imitation Learning
aka learning by demonstration

- Sequential decision making problem
  - At each point in time t
    - Receive input information x_t
    - Take action a_t
    - Suffer loss l_t
    - Move to next time step until time T
- Goal
  - learn a **policy** function $f(x_t) = y_t$
  - That minimizes expected **total** loss over all **trajectories** enabled by f

$$\boldsymbol{\tau} = \boldsymbol{x}_1\,,\ \underbrace{a_1}_{=f(\boldsymbol{x}_1)},\ell_1\,,\ \boldsymbol{x}_2\,,\ \underbrace{a_2}_{=f(\boldsymbol{x}_2)},\ell_2\,,\ \ldots\,,\ \boldsymbol{x}_T\,,\ \underbrace{a_T}_{=f(\boldsymbol{x}_T)},\ell_T$$

Supervised Imitation Learning

Algorithm 43 SupervisedImitationTrain($\mathcal{A}, \tau_1, \tau_2, \ldots, \tau_N$)

$D \leftarrow \left\langle (x, a) \ : \ \forall n \ , \ \forall (x, a, \ell) \in \tau_n \right\rangle$ // collect all observation/action pairs

return $\mathcal{A}(D)$ // train multiclass classifier on D

Algorithm 44 SupervisedImitationTest(f)

for $t = 1 \ldots T$ **do**

  $x_t \leftarrow$ current observation

  $a_t \leftarrow f(x_t)$ // ask policy to choose an action

  take action a_t

  $\ell_t \leftarrow$ observe instantaneous loss

end for

return $\sum_{t=1}^{T} \ell_t$ // return total loss

Supervised Imitation Learning

Algorithm 43 SUPERVISEDIMITATIONTRAIN($\mathcal{A}, \tau_1, \tau_2, \ldots, \tau_N$)

1: $D \leftarrow \langle (x, a) \ : \ \forall n \ , \ \forall (x, a, \ell) \in \tau_n \rangle$ // collect all observation/acti n pairs
2: **return** $\mathcal{A}(D)$ // t in multiclass er on D

Algorithm 44 SUPERVISEDIMITA

1: **for** $t = 1 \ldots T$ **do**
2: $x_t \leftarrow$ current obse
3: $a_t \leftarrow f(x_t)$
4: take actio
5: $\ell_t \leftarrow$ observe inst
6: **end for**
7: **return** $\sum_{t=1}^{T} \ell_t$ // return total loss

Problem with supervised approach: Compounding error

How can we train system to make better predictions off the expert path?

- We want a policy f that leads to good performance in configurations that f encounters
- A chicken and egg problem
  - Can be addressed by iterative approach

DAGGER: simple & effective imitation learning via Data AGGregation

Algorithm 45 DaggerTrain($\mathcal{A}$, MaxIter, N, expert)

1: $\langle \boldsymbol{\tau}_n^{(0)} \rangle_{n=1}^N \leftarrow$ run the expert N many times
2: $D_0 \leftarrow \left\langle (\boldsymbol{x}, a) \ : \ \forall n \ , \ \forall (\boldsymbol{x}, a, \ell) \in \boldsymbol{\tau}_n^{(0)} \right\rangle$ // collect all pairs (same as supervised)
3: $f_0 \leftarrow \mathcal{A}(D_0)$ // train initial policy (multiclass classifier) on D_0
4: **for** $i = 1 \ldots MaxIter$ **do**
5: $\quad \langle \boldsymbol{\tau}_n^{(i)} \rangle_{n=1}^N \leftarrow$ run policy f_{i-1} N-many times // trajectories by f_{i-1}
6: $\quad D_i \leftarrow \left\langle (\boldsymbol{x}, \text{expert}(\boldsymbol{x})) \ : \ \forall n \ , \ \forall (\boldsymbol{x}, a, \ell) \in \boldsymbol{\tau}_n^{(i)} \right\rangle$ // collect data set
// observations $\boldsymbol{x}$ visited by f_{i-1}
// but actions according to the expert
7: $\quad f_i \leftarrow \mathcal{A}\left(\bigcup_{j=0}^{i} D_j \right)$ // train policy f_i on union of all data so far
8: **end for**
9: **return** $\langle f_0, f_1, \ldots, f_{\text{MaxIter}} \rangle$ // return collection of all learned policies

Requires interaction with expert!

When is DAGGER used in practice?

- Interaction with expert is not always possible
- Classic use case
  - Expert = slow algorithm
  - Use DAGGER to learn a faster algorithm that imitates expert
  - Example: game playing where expert = brute-force search in simulation mode
- But also structured prediction

Sequence labeling via imitation learning

$x =$ " monsters eat tasty bunnies "

$y =$ noun verb adj noun

- What is the "expert" here?
  - Given a loss function (e.g., Hamming loss)
  - Expert takes action that minimizes long-term loss

$$\text{expert}(\ell, y, \hat{y}) = \underset{a}{\text{argmin}}\, \text{best}(\ell, y, \hat{y} \circ a)$$

Output prefix at time t

Loss of best reachable output starting with prefix $\hat{y} \circ a$

- When expert can be computed exactly, it is called an **oracle**

- Key advantages
  - Can define features $\phi(x, \hat{y})$
  - No restriction to Markov features

Today's topics

- Improving on gold parse oracle
  - Research highlight: [Goldberg & Nivre, 2012]
- Imitation learning for structured prediction
  - CIML ch 18