From Dependency Parsing to Imitation Learning

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\textsuperscript{1} Joakim Nivre\textsuperscript{1,2}
(1) Google Inc.
(2) Uppsala University
yogo@google.com, joakim.nivre@lingfil.uu.se
Exercise: which of these transition sequences produces the gold tree on the left?
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: \( t \leftarrow \text{LEFT-ARC}_l \)
3: else if \( c = (\sigma|i, j|\beta, A) \) and \((i, l, j) \in A_{\text{gold}} \) then
4: \( t \leftarrow \text{RIGHT-ARC}_l \)
5: else if \( c = (\sigma|i, j|\beta, A) \) and \( \exists k [k < i \land \exists l [(k, l, j) \in A_{\text{gold}} \lor (j, l, k) \in A_{\text{gold}}]] \) then
6: \( t \leftarrow \text{REDUCE} \)
7: else
8: \( t \leftarrow \text{SHIFT} \)
9: 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:     $t \leftarrow \text{LEFT-ARC}_l$
3: else if $c = (\sigma | i, j | \beta, A)$ and $(i, l, j) \in A_{\text{gold}}$ then
4:     $t \leftarrow \text{RIGHT-ARC}_l$
5: else if $c = (\sigma | i, j | \beta, A)$ and $\exists k [k < i \land \exists l [(k, l, j) \in A_{\text{gold}} \lor (j, l, k) \in A_{\text{gold}}]]$ then
6:     $t \leftarrow \text{REDUCE}$
7: else
8:     $t \leftarrow \text{SHIFT}$
9: return $t$

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 Goldberg¹ Joakim Nivre¹,²
(1) Google Inc.
(2) Uppsala University
yogo@google.com, joakim.nivre@lingfil.uu.se
At test time, suppose the 4\textsuperscript{th} transition predicted is SHIFT instead of RA\textsubscript{OBJ}.

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 Goldberg¹  Joakim Nivre¹,²
(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: $w \leftarrow 0$
2: for $i = 1 \rightarrow$ ITERATIONS do
3: for sentence $x$ with gold tree $G_{\text{gold}}$ in corpus do
4: $c \leftarrow c_s(x)$
5: while $c$ is not terminal do
6: $t_p \leftarrow \arg \max_t w \cdot \phi(c, t)$
7: $\text{ZERO}_\text{COST} \leftarrow \{t| o(t; c, G_{\text{gold}}) = \text{true}\}$
8: $t_o \leftarrow \arg \max_{t \in \text{ZERO}_\text{COST}} w \cdot \phi(c, t)$
9: if $t_p \not\in \text{ZERO}_\text{COST}$ then
10: $w \leftarrow w + \phi(c, t_o) - \phi(c, t_p)$
11: $t_n \leftarrow \text{CHOOSE}_\text{NEXT}(I, t_p, \text{ZERO}_\text{COST})$
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: \( w \leftarrow 0 \)
2: \textbf{for} \( I = 1 \rightarrow \text{ITERATIONS} \) \textbf{do}
3: \textbf{for} sentence \( x \) with gold tree \( G_{\text{gold}} \) in corpus \textbf{do}
4: \( c \leftarrow c_s(x) \)
5: \textbf{while} \( c \) is not terminal \textbf{do}
6: \( t_p \leftarrow \arg \max_t \ w \cdot \phi(c, t) \)
7: \( \text{ZERO\_COST} \leftarrow \{ t \mid o(t; c, G_{\text{gold}}) = \text{true} \} \)
8: \( t_o \leftarrow \arg \max_{t \in \text{ZERO\_COST}} \ w \cdot \phi(c, t) \)
9: \textbf{if} \( t_p \notin \text{ZERO\_COST} \) \textbf{then}
10: \( w \leftarrow w + \phi(c, t_o) - \phi(c, t_p) \)
11: \( t_n \leftarrow \text{CHOOSE\_NEXT}(I, t_p, \text{ZERO\_COST}) \)
12: \( c \leftarrow t_n(c) \)
13: \textbf{return} \( w \)

1: \textbf{function} \text{CHOOSE\_NEXT\_AMB}(I, t, \text{ZERO\_COST})
2: \textbf{if} \( t \in \text{ZERO\_COST} \) \textbf{then}
3: \textbf{return} \( t \)
4: \textbf{else}
5: \textbf{return} \text{RANDOM\_ELEMENT}(\text{ZERO\_COST})

1: \textbf{function} \text{CHOOSE\_NEXT\_EXP}(I, t, \text{ZERO\_COST})
2: \textbf{if} \( I > k \) and \text{RAND()} > p \textbf{then}
3: \textbf{return} \( t \)
4: \textbf{else}
5: \textbf{return} \text{CHOOSE\_NEXT\_AMB}(I, t, \text{ZERO\_COST})
Defining the cost of a transition

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

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

• 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 $f$

$$\tau = x_1, a_1, l_1, x_2, a_2, l_2, \ldots, x_T, a_T, l_T$$

$$= f(x_1) = f(x_2) = f(x_T)$$
Supervised Imitation Learning

Algorithm 43 \textsc{SupervisedImitationTrain}(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/action pairs
2: \textbf{return} \ A(D)  // train multiclass classifier on D

Algorithm 44 \textsc{SupervisedImitationTest}(f)

1: \textbf{for} \( t = 1 \ldots T \) \textbf{do}
2: \hspace{1em} \( x_t \leftarrow \) current observation
3: \hspace{1em} \( a_t \leftarrow f(x_t) \) // ask policy to choose an action
4: \hspace{1em} take action \( a_t \)
5: \hspace{1em} \( \ell_t \leftarrow \) observe instantaneous loss
6: \textbf{end for}
7: \textbf{return} \( \sum_{t=1}^{T} \ell_t \) // return total loss
Supervised Imitation Learning

Problem with supervised approach: Compounding error

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

1: \( D \leftarrow \langle (x, a) : \forall n, (x, a, \ell) \in \tau_n \rangle \) \hspace{1em} // collect all observation/action pairs
2: \textbf{return } A(D) \hspace{1em} // train multiclass classifier on \( D \)

Algorithm 44 \textsc{SupervisedImitationTest}(x, A)

1: \textbf{for } t = 1 \ldots T \textbf{ do}
2: \( x_t \leftarrow \text{current observation} \)
3: \( a_t \leftarrow f(x_t) \)
4: \text{take action} \( a_t \)
5: \( \ell_t \leftarrow \text{observe instant loss} \)
6: \textbf{end for}
7: \textbf{return } \sum_{t=1}^{T} \ell_t \hspace{1em} // return total loss
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**(A, MaxIter, N, expert)

1. \( \langle \tau_n^{(0)} \rangle_{n=1}^N \leftarrow \) run the expert N many times
2. \( D_0 \leftarrow \langle (x, a) : \forall n, \forall (x, a, \ell) \in \tau_n^{(0)} \rangle \)  // collect all pairs (same as supervised)
3. \( f_0 \leftarrow A(D_0) \)  // train initial policy (multiclass classifier) on \( D_0 \)
4. **for** \( i = 1 \ldots \text{MaxIter} \) **do**
5. \( \langle \tau_n^{(i)} \rangle_{n=1}^N \leftarrow \) run policy \( f_{i-1} \) N-many times  // trajectories by \( f_{i-1} \)
6. \( D_i \leftarrow \langle (x, \text{expert}(x)) : \forall n, \forall (x, a, \ell) \in \tau_n^{(i)} \rangle \)  // collect data set
   // observations \( x \) visited by \( f_{i-1} \)
   // but actions according to the expert
7. \( f_i \leftarrow A \left( \bigcup_{j=0}^i D_j \right) \)  // 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

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

• 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

$x = "\text{monsters eat tasty bunnies}"

y = \text{noun \ verb \ adj \ noun}$
Today’s topics

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

• Imitation learning for structured prediction
  • CIML ch 18