Transition Based Parsing: Learning

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle.**
- An oracle can be approximated by a **classifier.**
- A classifier can be trained using **treebank data.**

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle.**
- An oracle can be approximated by a **classifier.**
- A classifier can be trained using **treebank data.**

Learning Problem

- Approximate a function from **configurations, represented by feature vectors** to **transitions,** given a training set of gold standard **transition sequences.**

Classifier-Based Parsing

Data-driven deterministic parsing:

- Deterministic parsing requires an **oracle.**
- An oracle can be approximated by a **classifier.**
- A classifier can be trained using **treebank data.**

Learning Problem

- Approximate a function from **configurations, represented by feature vectors** to **transitions,** given a training set of gold standard **transition sequences.**

Three issues

- How to represent configurations by feature vectors?
- How to derive training data from treebanks?
- How to learn classifiers?

How to represent configurations by feature vectors?

- Configuration means: [ ]$_S$ [ ]$_B$ Arcs { }
- Convert configuration to some feature vector : f(c, t)
- f(c, t) $\rightarrow$ define certain conditions over stack, buffer and arcs

Feature Models

- A feature representation f (c) of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - Structural context (parents, children, siblings in G)

Feature Models

- A feature representation f (c) of a configuration c is a vector of simple features $f_i(c)$.

Typical Features

- Nodes:
  - Target nodes (top of S, head of B)
  - Linear context (neighbors in S and B)
  - Structural context (parents, children, siblings in G)
- Attributes:
  - Word form (and lemma)
  - Part-of-speech (and morpho-syntactic features)
  - Dependency type (if labeled)
  - Distance (between target tokens)

Deterministic Parsing

- At runtime, given a sentence S: $w_1,\ldots,w_n$
- Initial Configuration: $[\ \ ]_S\ \ [\ \ ]_B$ Arcs { }

Using classifier at run-time

PARSE($w_1,\ldots,w_n$)

1. $c \leftarrow([\ ]_S, [w_1,\ldots,w_n]_B, \{\ \})$
2. **while** $B_C \neq []$
3. $t^* \leftarrow \text{argmax}_t\ w \cdot f(c, t)$
4. $c \leftarrow t^*(c)$
5. **return** $T = (\{w_1,\ldots,w_n\}, A_c)$

To guide the parser, a linear classifier can be used:
$t^* = \text{argmax}_t\ w \cdot f(c, t)$
Weight vector w learned from treebank data.

Training data: For learning weights

- Training instances have the form (f (c), t),where
  - f (c) is a feature representation of a configuration c,
  - t is the correct transition out of c (i.e., o(c) = t), where o is an oracle . An oracle is set of configurations and set of transitions that we have taken.
- Given a dependency treebank, we can sample the oracle function o as follows:
  - For each sentence x with gold standard dependency graph G_x, construct a transition sequence $C_{0,m} = (c_0, c_1,\ldots,c_m)$ such that

    $c_0 = c_s(x)$,

    $G_{cm} = G_x$
  - For each configuration $c_i (i < m)$, we construct a training instance $(f(c_i), t_i)$, where $t_i (c_i) = c_{i+1}$.

Standard Oracle for Arc-Eager Parsing

o(c, T) =

- **Left-Arc** if $top(S_c) \leftarrow first(B_c)$ in T
- **Right-Arc** if $top(S_c) \rightarrow first(B_c)$ in T
- **Reduce** if $\exists w < top(S_c) : w \leftrightarrow first(B_c)$ in T
- **Shift** otherwise

Online Learning with an Oracle

LEARN($\{T_1, \ldots, T_N\}$)

1 $w \leftarrow 0.0$
2 **for** i in $1..K$
3 **for** j in $1..N$
4 $c \leftarrow ([]_S, [w_1, \ldots, w_{n_j}]_B, \{\})$
5 **while** $B_c \neq []$
6 $t^* \leftarrow \arg\max_t w.f(c,t)$
7 $t_o \leftarrow o(c, T_i)$
8 **if** $t^* \neq t_o$
9 $w \leftarrow w + f(c,t_o) - f(c,t^*)$
10 $c \leftarrow t_o(c)$
11 **return** w

- Oracle o(c, T_i) returns the optimal transition of c and Ti

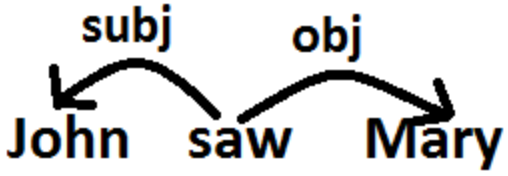
Example: **Consider the sentence, 'John saw Mary'**

- **Draw a dependency graph** for this sentence.
- Assume that you are learning a classifier for the data-driven deterministic parsing and the above sentence is a gold-standard parse in your training data. You are also given that John and Mary are 'Nouns', while the POS tag of saw is 'Verb'. Assume that your features correspond to the following conditions:
  - The stack is empty
  - Top of stack is Noun and Top of buffer is Verb
  - Top of stack is Verb and Top of buffer is Noun

  Initialize the weights of all your features to 5.0, except that in all of the above cases, you give a weight of 5.5 to Left-Arc. **Define your feature vector and the initial weight vector.**
- Use this gold standard parse during online learning and report the **weights** after completing one full iteration of Arc-Eager parsing over this sentence.

Solution:

1) Dependency graph :



2) Define feature vector and initial weight vectors

How many conditions over initial configuration -3

These conditions have to be checked for all the 4 transitions –LA, RA, Reduce and shift

So, size of feature vector: 3 * 4 =12

f(c, t) = [c_1LA, c_2LA, c_3LA, c_1RA, c_2RA, c_3RA, (c_1,RE), (c_2,RE), (c_3,RE), (c_1,SH), (c_2,SH), (c_3, SH)]

Now, the initial weight vector is :

W= [5.5, 5.5, 5.5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

How do we learn weights?

- Use gold-standard parse. Leran weights using Arc-eager parsing

Initial configuration C : $[\ ]_S\ [\ \text{John, saw, Mary}]_B\ \{\ \}$

From Oracle, whenever stack is empty we choose SHIFT transition, say t0 = SH.

From classifier, we choose :

$t^* \leftarrow \text{argmax}_t\ w\ .\ f\ (c, t)$

1) Condition is top of stack empty and left Arc

$f(c_1, \text{LA}) = [1,0,0,0,0,0,0,0,0,0,0,0,0]$

$w*f(c_1, \text{LA}) = [5.5,0,0,0,0,0,0,0,0,0,0,0,0]$

2) Condition is top of stack empty and Right Arc

$f(c_1, RA) = [0,0,0,1,0,0,0,0,0,0,0,0]$

$w*f(c_1, RA) = [0,0,0,5,0,0,0,0,0,0,0,0]$

3) Condition is top of stack empty and Reduce

$f(c_1, RE) = [0,0,0,0,0,0,1,0,0,0,0,0]$

$w*f(c_1, RE) = [0,0,0,0,0,0,5,0,0,0,0,0]$

4) Condition is top of stack empty and Shift

$f(c_1, SH) = [0,0,0,0,0,0,0,0,0,1,0,0]$

$w*f(c_1, SH) = [0,0,0,0,0,0,0,0,0,5,0,0]$

Now,

$t^* \leftarrow \text{argmax}_t\ w \cdot f(c, t) = LA$

t0 = SH (as per oracle)

How do we learn weights:

If $(t^* \neq t0)$, update

$w' = w + f(c, t0) - f(c, t^*)$

W= [5.5, 5.5, 5.5, 5, 5, 5, 5, 5, 5, 5, 5, 5]

f(c, t0) = [0,0,0,0,0,0,0,0,0,1,0,0]

f(c, t*) = [1,0,0,0,0,0,0,0,0,0,0,0]

w' = [4.5, 5.5, 5.5, 5, 5, 5, 5, 5, 5, 6, 5, 5]

Now, this weight vector will be used for the next configuration.

Next C = [John]$_S$ [saw, Mary]$_B$ { }

From Oracle, find t0 = LA transition

From classifier, find t* =

5) Condition is Top of stack is Noun and Top of buffer is Verb and left Arc

f(c_2, LA) = [0,1,0,0,0,0,0,0,0,0,0,0]

w*f(c_2, LA) = [0,5.5,0,0,0,0,0,0,0,0,0,0]

6) Condition is Top of stack is Noun and Top of buffer is Verb and right Arc

$f(c_2, RA) = [0,0,0,0,1,0,0,0,0,0,0,0]$

$w*f(c_2, RA) = [0,0,0,0,5,0,0,0,0,0,0,0]$

7) Condition is Top of stack is Noun and Top of buffer is Verb and Reduce

$f(c_2, RE) = [0,0,0,0,0,0,0,1,0,0,0,0]$

$w*f(c_2, RE) = [0,0,0,0,0,0,0,5,0,0,0,0]$

8) Condition is Top of stack is Noun and Top of buffer is Verb and Shift

$f(c_2, SH) = [0,0,0,0,0,0,0,0,0,0,1,0]$

$w*f(c_2, SH) = [0,0,0,0,0,0,0,0,0,0,5,0]$

$t* \leftarrow \text{argmax}_t\ w \,.\, f(c, t) = LA$

t0 = LA (as per oracle)

If (t*=t0), no update w'' = w'

Next C = []S [saw, Mary]B {John <- saw}

From Oracle, find t0 = SH transition

From classifier, find t* =

w' = [4.5, 5.5, 5.5, 5, 5, 5, 5, 5, 5, 6, 5, 5]

....