

NLP

Introduction to NLP

Dependency Parsing 2

Transition-based Dependency Parsing

Transition-Based Parsing

- Similar to shift–reduce
- Produces a single (projective) tree
- Data structures
 - Stack of partially processed (unattached) words
 - Input buffer
 - Set of dependency arcs
 - Attach the word on the top of the stack to the word at the current position in the buffer (or in the other direction)

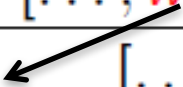

Transition-Based Parsing

- Initial configuration
 - Stack (including the root token w_0)
 - Buffer (sentence)
 - Arcs (empty)
- Goal configuration
 - Stack (empty)
 - Buffer (empty)
 - Arcs (complete tree)

MaltParser (Nivre 2008)

- The reduce operations combine an element from the stack and one from the buffer
- Arc-standard parser
 - The actions are shift, left-arc, right-arc
- Arc-eager parser
 - The actions are shift, reduce, left-arc, right-arc

(Arc-Eager) MaltParser Actions

Shift	$\frac{[\dots]_S \quad [w_i, \dots]_Q}{[\dots, w_i]_S \quad [\dots]_Q}$
Reduce	$\frac{[\dots, w_i]_S \quad [\dots]_Q \quad \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [\dots]_Q}$
Left-Arc _r	$\frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_i}{[\dots]_S \quad [w_j, \dots]_Q \quad w_i \xleftarrow{r} w_j}$ 
Right-Arc _r	$\frac{[\dots, w_i]_S \quad [w_j, \dots]_Q \quad \neg \exists w_k : w_k \rightarrow w_j}{[\dots, w_i, w_j]_S \quad [\dots]_Q \quad w_i \xrightarrow{r} w_j}$ 

[Example from Nivre and Kuebler]

Transition Configuration

	([ROOT],	[Economic, . . . , .],	\emptyset)
SH \Rightarrow	([ROOT, Economic],	[news, . . . , .],	\emptyset)
LA _{ATT} \Rightarrow	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$
SH \Rightarrow	([ROOT, news],	[had, . . . , .],	A_1)
LA _{SBJ} \Rightarrow	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$
RA _{PRED} \Rightarrow	([ROOT, had],	[little, . . . , .],	$A_3 = A_2 \cup \{(root, PRED, had)\}$
SH \Rightarrow	([ROOT, had, little],	[effect, . . . , .],	A_3)
LA _{ATT} \Rightarrow	([ROOT, had],	[effect, . . . , .],	$A_4 = A_3 \cup \{(effect, ATT, little)\}$
RA _{OBJ} \Rightarrow	([ROOT, had, effect],	[on, . . . , .],	$A_5 = A_4 \cup \{(had, OBJ, effect)\}$
RA _{ATT} \Rightarrow	([ROOT, . . . on],	[financial, markets, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$
SH \Rightarrow	([ROOT, . . . , financial],	[markets, .],	A_6)
LA _{ATT} \Rightarrow	([ROOT, . . . on],	[markets, .],	$A_7 = A_6 \cup \{(markets, ATT, financial)\}$
RA _{PC} \Rightarrow	([ROOT, . . . , markets],	[.],	$A_8 = A_7 \cup \{(on, PC, markets)\}$
RE \Rightarrow	([ROOT, . . . , on],	[.],	A_8)
RE \Rightarrow	([ROOT, . . . , effect],	[.],	A_8)
RE \Rightarrow	([ROOT, had],	[.],	A_8)
RA _{PU} \Rightarrow	([ROOT, . . . , .],	[.],	$A_9 = A_8 \cup \{(had, PU, .)\}$

Figure 3.7: Arc-eager transition sequence for the English sentence in figure 1.1 (LA_r = LEFT-ARC_r, RA_r = RIGHT-ARC_r, RE = REDUCE, SH = SHIFT).

[Example from Kuebler, McDonald, Nivre]

Example

- Example: “People want to be free”

- [ROOT] [People, want, to, be, free] \emptyset
- Shift [ROOT, People] [want, to, be, free]
- LA_{nsubj} [ROOT] [want, to, be, free] $A_1 = \{nsubj(want, people)\}$
- RA_{root} [ROOT, want] [to, be, free] $A_2 = A_1 \cup \{root(ROOT, want)\}$

- Characteristics

- The next action is chosen locally using a classifier (e.g. SVM)
- There is no search
- The final list of arcs is returned as the dependency tree
- Trained on a dependency treebank

- Very fast method

Parsing

- Oracle-based
 - Assuming an oracle, parsing is deterministic
- In practice
 - Approximate the oracle with a classifier
 - $o(c) = \operatorname{argmax}_t \mathbf{w} \cdot \mathbf{f}(c, t)$

```
Parse( $w_1, \dots, w_n$ )  
1   $c \leftarrow ([ ]_S, [0, 1, \dots, n]_B, \{ \})$   
2  while  $B_c \neq [ ]$   
3       $t \leftarrow o(c)$   
4       $c \leftarrow t(c)$   
5  return  $G = (\{0, 1, \dots, n\}, A_c)$ 
```

[Example from McDonald and Nivre]

Greedy Transition-Based Parsing

- Beam search with $q=1$
- Score is computed using a linear model
- Because of the greedy property, errors can propagate

```
Parse (sent =  $w_1 \dots w_n$ )  
   $c = cs(sent)$   
  While  $c$  is not in  $C_t$   
     $t^* = \operatorname{argmax}_t \text{score}(c, t)$   
     $c = t^*(c)$   
  Return  $Gc$ 
```

Feature Model

Table 3.1: Feature model for transition-based parsing.

f_i	Address	Attribute
1	STK[0]	FORM
2	BUF[0]	FORM
3	BUF[1]	FORM
4	LDEP(STK[0])	DEPREL
5	RDEP(STK[0])	DEPREL
6	LDEP(BUF[0])	DEPREL
7	RDEP(BUF[0])	DEPREL

[Example from Kuebler, McDonald, Nivre]

Feature Vectors

$f(c_0)$	=	(ROOT	Economic	news	NULL	NULL	NULL	NULL)
$f(c_1)$	=	(Economic	news	had	NULL	NULL	NULL	NULL)
$f(c_2)$	=	(ROOT	news	had	NULL	NULL	ATT	NULL)
$f(c_3)$	=	(news	had	little	ATT	NULL	NULL	NULL)
$f(c_4)$	=	(ROOT	had	little	NULL	NULL	SBJ	NULL)
$f(c_5)$	=	(had	little	effect	SBJ	NULL	NULL	NULL)
$f(c_6)$	=	(little	effect	on	NULL	NULL	NULL	NULL)
$f(c_7)$	=	(had	effect	on	SBJ	NULL	ATT	NULL)
$f(c_8)$	=	(effect	on	financial	ATT	NULL	NULL	NULL)
$f(c_9)$	=	(on	financial	markets	NULL	NULL	NULL	NULL)
$f(c_{10})$	=	(financial	markets	.	NULL	NULL	NULL	NULL)
$f(c_{11})$	=	(on	markets	.	NULL	NULL	ATT	NULL)
$f(c_{12})$	=	(effect	on	.	ATT	NULL	NULL	ATT)
$f(c_{13})$	=	(had	effect	.	SBJ	NULL	ATT	ATT)
$f(c_{14})$	=	(ROOT	had	.	NULL	NULL	SBJ	OBJ)
$f(c_{15})$	=	(had	.	NULL	SBJ	OBJ	NULL	NULL)
$f(c_{16})$	=	(ROOT	had	NULL	NULL	NULL	SBJ	PU)
$f(c_{17})$	=	(NULL	ROOT	NULL	NULL	NULL	NULL	PRED)
$f(c_{18})$	=	(ROOT	NULL	NULL	NULL	PRED	NULL	NULL)

Figure 3.5: Feature vectors for the configurations in figure 3.2.

[Example from Kuebler, McDonald, Nivre]

Complexity

- Arc-eager is $O(n^3)$ – like Eisner
- Arc-standard is $O(n^5)$

NLP