Transition-based dependency parsing

Transition systems for dependency parsing

Transitions are between **configurations** that are represented as triples $C \rightarrow (\sigma, \beta, A)$, where σ is the stack, β is the input buffer, and A is the list of arcs that have been created.

Transition systems:

- Arc-standard
- Arc-eager

The arc-standard system

The arc-standard system is closely related to the shift-reduce algorithm for phrase structure parsing, with the difference being that the REDUCE action is split into two actions, ARC-LEFT and ARC-RIGHT, depending on whether the head is on the left or right.

- SHIFT $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$
- ARC-LEFT $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, j|\beta, A \oplus j \rightarrow i)$
- ARC-RIGHT $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, i|\beta, A \oplus i \xrightarrow{r} j)$

Arc-standard derivation

	σ	β	action	arc added to $\mathcal{A}$
1.	[ROOT]	*they like bagels with lox*	SHIFT	
2.	[ROOT, *they*]	*like bagels with lox*	ARC-LEFT	(*they* $\leftarrow$ *like*)
3.	[ROOT]	*like bagels with lox*	SHIFT	
4.	[ROOT, *like*]	*bagels with lox*	SHIFT	
5.	[ROOT, *like, bagels*]	*with lox*	SHIFT	
6.	[ROOT, *like, bagels, with*]	*lox*	ARC-LEFT	(*with* $\leftarrow$ *lox*)
7.	[ROOT, *like, bagels*]	*lox*	ARC-RIGHT	(*bagels* $\rightarrow$ *lox*)
8.	[ROOT, *like*]	*bagels*	ARC-RIGHT	(*like* $\rightarrow$ *bagels*)
9.	[ROOT]	*like*	ARC-RIGHT	(ROOT $\rightarrow$ *like*)
10.	[ROOT]	$\varnothing$	DONE	

Table 11.2: Arc-standard derivation of the unlabeled dependency parse for the input *they like bagels with lox*.

Arc-eager transition system

Arc-eager dependency parsing changes the ARC-RIGHT action so that right dependents can be attached before all of their dependents have been found. Rather than removing the modifier from both the buffer and stack, the ARC-RIGHT action pushes the modifier on to the stack of the head. Two additional changes are necessary:

- A precondition is required to ensure that the ARC-LEFT action cannot be applied when the top element on the stack already has a parent A.
- A new REDUCE action is introduced, which can remove elements from the stack if they have a parent A:
  $(\sigma|i, \beta, A) \Rightarrow (\sigma, \beta, A)$

Arc-eager derivation

	σ	β	action	arc added to $\mathcal{A}$
1.	[ROOT]	*they like bagels with lox*	SHIFT	
2.	[ROOT, *they*]	*like bagels with lox*	ARC-LEFT	(*they* $\leftarrow$ *like*)
3.	[ROOT]	*like bagels with lox*	ARC-RIGHT	(ROOT $\rightarrow$ *like*)
4.	[ROOT, *like*]	*bagels with lox*	ARC-RIGHT	(*like* $\rightarrow$ *bagels*)
5.	[ROOT, *like, bagels*]	*with lox*	SHIFT	
6.	[ROOT, *like, bagels, with*]	*lox*	ARC-LEFT	(*with* $\leftarrow$ *lox*)
7.	[ROOT, *like, bagels*]	*lox*	ARC-RIGHT	(*bagels* $\rightarrow$ *lox*)
8.	[ROOT, *like, bagels, lox*]	$\varnothing$	REDUCE	
9.	[ROOT, *like, bagels*]	$\varnothing$	REDUCE	
10.	[ROOT, *like*]	$\varnothing$	REDUCE	
11.	[ROOT]	$\varnothing$	DONE	

Table 11.3: Arc-eager derivation of the unlabeled dependency parse for the input *they like bagels with lox*.

Oracle-based training

An **oracle** is a function that maps a parse tree into an action sequence. Given such an oracle, a dependency treebank can be converted into a set of action sequences $\{A^{(i)}\}_{i=1}^{N}$. The parser can be trained by stepping through the oracle action sequences, and optimizing on an classification-based objective that rewards the oracle action. A commonly used objective is to maximize the conditional likelihood (or minimize the negative log conditional likelihood).

$$P(a|c, \boldsymbol{w}) = \frac{\exp \psi(a, c, \boldsymbol{w}; \boldsymbol{\theta})}{\sum_{a' \in \mathcal{A}(c)} exp\psi(a', c, \boldsymbol{w}; \boldsymbol{\theta})}$$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmax}} \sum_{i=1}^{N} \sum^{|A^{(i)}|} \log P(a_t^{(i)}|c_t^{(i)}, \boldsymbol{w})$$