Transition-based dep

## Transition systems for dependency parsing



Transitions are between **configurations** ed as triples $C \rightarrow (\sigma, \beta, A)$, where $\sigma$ is the stack, uffer, and $A$ is the list of arc



Transition system

- Arc-standard
- Arc-eager

## The arc-standard system



The arc-standard system is closely related to the shi
algorithm for phrase structure parsing, with the di
that the REDUCE action is split into two action -LEFT and
ARC-RIGHT, dep t.



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

## Arc-standard derivation



| | $\sigma$ | $\beta$ | action | arc added to $\mathcal{A}$ |
|---|---|---|---|---|
| 1. | [ROOT] | *they like bagels with lox* | | |
| 2. | [ROOT, *they*] | *like bagels with lox* | SH | |
| 3. | [ROOT] | *like bagels with lox* | | |
| 4. | [ROOT, *like*] | | | |
| 5. | [ROOT, *like, ba* | | | |
| 6. | [ROOT, *like, ba* | | FT | (*with* $\leftarrow$ *lox*) |
| 7. | [ROOT, *like, bagels*] | *lox* | | (*bagels* $\rightarrow$ *lox*) |
| 8. | [ROOT, *like*] | *bagels* | | (*like* $\rightarrow$ *bagels*) |
| 9. | [ROOT] | *like* | | (ROOT $\rightarrow$ *like*) |
| 10. | [ROOT] | $\varnothing$ | ONE | |

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 removi
from both the buffer and stack, the AR pushes the modifier on to the sta
necessary:



- A precondition is required to ensure that t RC-LEFT action cannot be applied when the top ele 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



| | $\sigma$ | $\beta$ | action | arc added to $\mathcal{A}$ |
|---|---|---|---|---|
| 1. | [ROOT] | *they like bagels with lox* | | |
| 2. | [ROOT, *they*] | *like bagels with lox* | LA | (*they* $\leftarrow$ *like*) |
| 3. | [ROOT] | *like bagels with lox* | ROOT | (ROOT $\rightarrow$ *like*) |
| 4. | [ROOT, *like*] | | RA | (*like* $\rightarrow$ *bagels*) |
| 5. | [ROOT, *like*, *bag* | | | |
| 6. | [ROOT, *like*, *bag* | | RA | (*with* $\leftarrow$ *lox*) |
| 7. | [ROOT, *like*, *bag* | | RA | (*bagels* $\rightarrow$ *lox*) |
| 8. | [ROOT, *like*, *bagels*, *lox*] | $\varnothing$ | | |
| 9. | [ROOT, *like*, *bagels*] | $\varnothing$ | | |
| 10. | [ROOT, *like*] | $\varnothing$ | | |
| 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 functio
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 seq
optimizing on an classification-based objective t
oracle action. A commonly used objective is to maximize the conditional likelih
likelihood).



nal

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

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