Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism

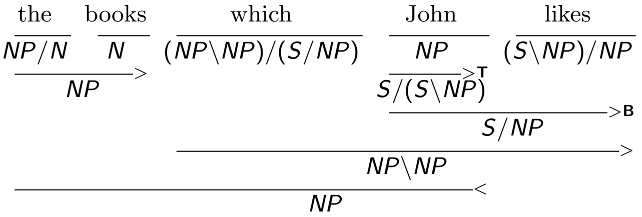
Combinatory Categorial Grammar (CCG)

$$
\begin{array}{ccccc}
\text{the} & \text{books} & \text{which} & \text{John} & \text{likes} \\
\overline{NP/N} & \overline{N} & \overline{(NP\backslash NP)/(S/NP)} & \overline{NP} & \overline{(S\backslash NP)/NP} \\
\multicolumn{2}{c}{\overline{\qquad\qquad}{>}} & & \overline{\qquad}{>}\mathbf{T} & \\
\multicolumn{2}{c}{NP} & & S/(S\backslash NP) & \\
 & & & \multicolumn{2}{c}{\overline{\qquad\qquad\qquad}{>}\mathbf{B}} \\
 & & & \multicolumn{2}{c}{S/NP} \\
 & & \multicolumn{3}{c}{\overline{\qquad\qquad\qquad\qquad\qquad}{>}} \\
 & & \multicolumn{3}{c}{NP\backslash NP} \\
\multicolumn{5}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad}{<}} \\
\multicolumn{5}{c}{NP}
\end{array}
$$

Combinatory Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures

Combinatory Categorial Grammar (CCG)

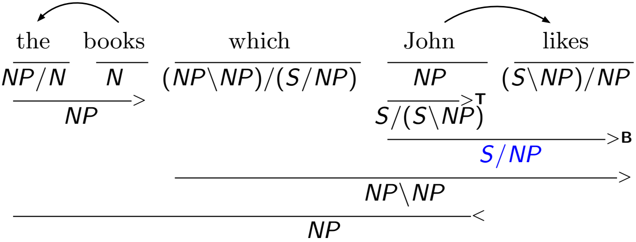


Combinatory Categorial Grammar (CCG)

the	books	which	John	likes
NP/N	N	$(NP\backslash NP)/(S/NP)$	NP	$(S\backslash NP)/NP$
$\xrightarrow{\quad\quad}{>}$ NP			$\xrightarrow{\quad}{>\mathbf{T}}$ $S/(S\backslash NP)$	
			$\xrightarrow{\quad\quad\quad}{>\mathbf{B}}$ S/NP	
		$\xrightarrow{\quad\quad\quad\quad}{>}$ $NP\backslash NP$		
$\xrightarrow{\quad\quad\quad\quad\quad}{<}$ NP				

$$\langle \mathit{the},\ NP/N_1,\ 1,\ \mathit{books}, \rangle$$

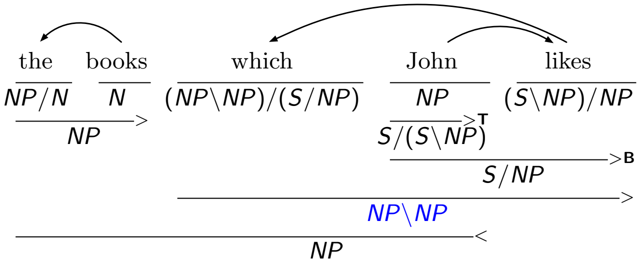
Combinatory Categorial Grammar (CCG)



$\langle the,\ NP/N_1,\ 1,\ books, \rangle$

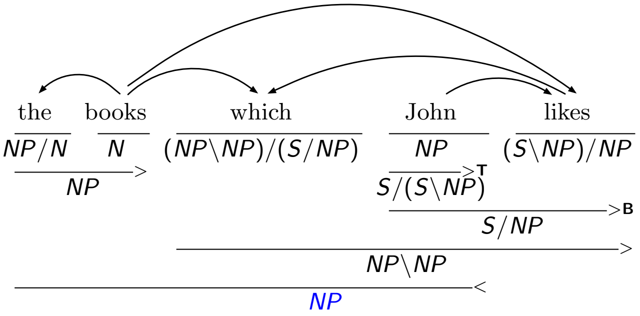
$\langle likes,\ (S\backslash NP_1)/NP_2,\ 1,\ John \rangle$

Combinatory Categorial Grammar (CCG)



$$\langle \textit{the},\ NP/N_{1},\ 1,\ \textit{books},\rangle$$
$$\langle \textit{likes},\ (S\backslash NP_{1})/NP_{2},\ 1,\ \textit{John}\rangle$$
$$\langle \textit{which},\ (NP/NP_{1})/(S/NP)_{2},\ 2,\ \textit{likes}\rangle$$

Combinatory Categorial Grammar (CCG)



$\langle \textit{the}, NP/N_1, 1, \textit{books}, \rangle$

$\langle \textit{likes}, (S\backslash NP_1)/NP_2, 1, \textit{John} \rangle$

$\langle \textit{which}, (NP/NP_1)/(S/NP)_2, 2, \textit{likes} \rangle$

$\langle \textit{which}, (NP/NP_1)/(S/NP)_2, 1, \textit{books} \rangle$

$\langle \textit{likes}, (S\backslash NP_1)/NP_2, 2, \textit{books} \rangle$

Combinatroy Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures

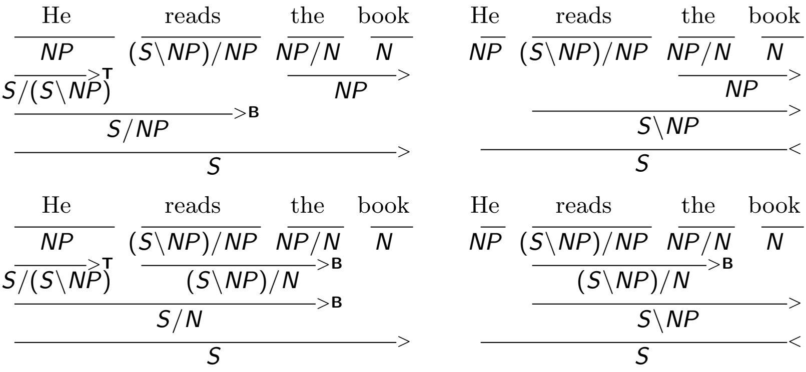
Combinatroy Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures
- Remains to be the most competitive formalism for recovering “deep” dependencies (from coordination, control, extraction etc.) [Rimell et al., 2009; Nivre et al., 2010]

Combinatroy Categorial Grammar (CCG)

- A lexicalized grammar formalism
- Is able to derive typed dependency structures
- Remains to be the most competitive formalism for recovering "deep" dependencies (from coordination, control, extraction etc.) [Rimell et al., 2009; Nivre et al., 2010]
- Exhibits "spurious" ambiguity (through the use of type-raising) in order to recover certain dependencies

Combinatroy Categorial Grammar (CCG)



$$\langle \textit{the},\ NP/N_1,\ 1,\ \textit{book},\ - \rangle$$
$$\langle \textit{reads},\ (S\backslash NP_1)/NP_2,\ 2,\ \textit{book},\ - \rangle$$
$$\langle \textit{reads},\ (S\backslash NP_1)/NP_2,\ 1,\ \textit{he},\ - \rangle$$

In general, exponentially many!

Motivation: Dependency Model

- Should we model the derivations or dependencies?
  - the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

- The derivation is just a "trace" of the semantic interpretation [Steedman, 2000]

Motivation: Dependency Model

- Should we model the derivations or dependencies?
  - the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

- The derivation is just a "trace" of the semantic interpretation [Steedman, 2000]

	[Clark et al., 2002]	C&C (dep)
Dep. Model	✓	✓
Deriv. Feats	✗	✓

Motivation: Dependency Model

- Should we model the derivations or dependencies?
  - the standard choice: the normal-form model [Hockenmaier, 2003; Clark and Curran 2007]

- The derivation is just a "trace" of the semantic interpretation [Steedman, 2000]

	[Clark et al., 2002]	C&C (dep)
Dep. Model	✓	✓
Deriv. Feats	✗	✓

  - an elegent solution to the spurious ambiguity problem
  - gold-standard data cheaper to obtain
  - optimizing for evaluation

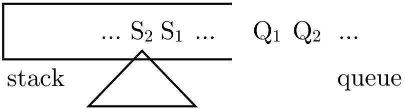
Motivation: Dependency Model

⇒ Open: Shift-Reduce CCG parsing with a Dependency Model

- Shift-Reduce fits with the incremental nature of CCG
  - linear vs. $\mathcal{O}(n^5)$ decoding
  - can include arbitrary features

	[Clark et al., 2002]	C&C (dep)	Z&C	this work
Shift-Reduce	✗	✗	✓	✓
Dep. Model	✓	✓	✗	✓
Deriv. Feats	✗	✓	✓	✓

Shift-Reduce CCG Parsing



- input: pos- and super-tagged words
- SHIFT: next lexical category from the queue
- REDUCE: the top two categories
- UNARY: type-raising or type-changing the top category

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	N/N N	plays Elianti	SHIFT

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	$N/N\ N$	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	N/N N	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	$N/N\ N$	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	$NP\ (S[dcl]\backslash NP)/NP$	Elianti	SHIFT

Shift-Reduce CCG Parsing

step	stack ($s_n, \ldots, s_1, s_0$)	queue ($q_0, q_1 \ldots, q_n$)	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	N/N N	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	NP $(S[dcl]\backslash NP)/NP$	Elianti	SHIFT
6	NP $(S[dcl]\backslash NP)/NP$ N		SHIFT

Shift-Reduce CCG Parsing

step	stack $(s_n, \ldots, s_1, s_0)$	queue $(q_0, q_1 \ldots, q_n)$	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	$N/N\ N$	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	$NP\ (S[dcl]\backslash NP)/NP$	Elianti	SHIFT
6	$NP\ (S[dcl]\backslash NP)/NP\ N$		SHIFT
7	$NP\ (S[dcl]\backslash NP)/NP\ NP$		UNARY

Shift-Reduce CCG Parsing

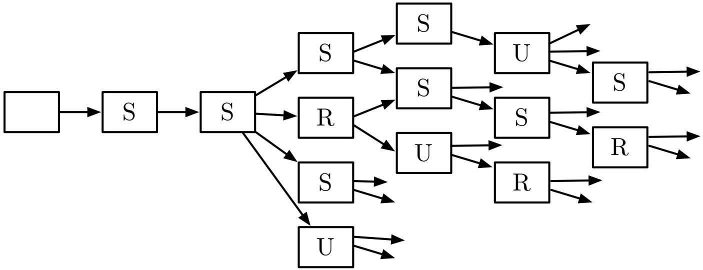
step	stack ($s_n, \ldots, s_1, s_0$)	queue ($q_0, q_1 \ldots, q_n$)	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	$N/N\ N$	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	$NP\ (S[dcl]\backslash NP)/NP$	Elianti	SHIFT
6	$NP\ (S[dcl]\backslash NP)/NP\ N$		SHIFT
7	$NP\ (S[dcl]\backslash NP)/NP\ NP$		UNARY
8	$NP\ S[dcl]\backslash NP$		REDUCE

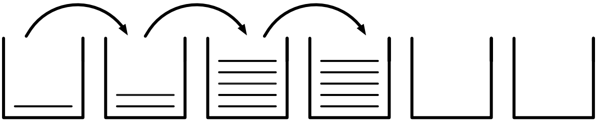
Shift-Reduce CCG Parsing

step	stack ($s_n, \ldots, s_1, s_0$)	queue ($q_0, q_1 \ldots, q_n$)	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	$N/N\ N$	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	$NP\ (S[dcl]\backslash NP)/NP$	Elianti	SHIFT
6	$NP\ (S[dcl]\backslash NP)/NP\ N$		SHIFT
7	$NP\ (S[dcl]\backslash NP)/NP\ NP$		UNARY
8	$NP\ S[dcl]\backslash NP$		REDUCE
9	$S[dcl]$		REDUCE

Beam-Search Decoding

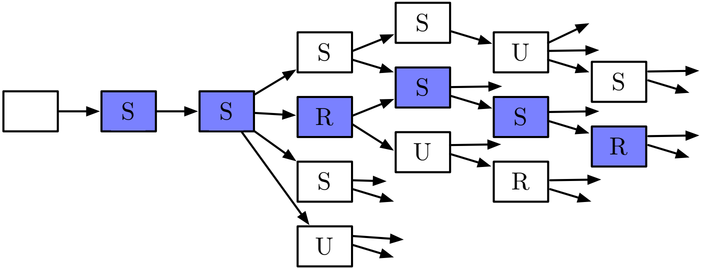
- Score of an item $\langle s, q \rangle \circ x = \mathbf{w} \cdot \phi(\langle s, q \rangle, x)$

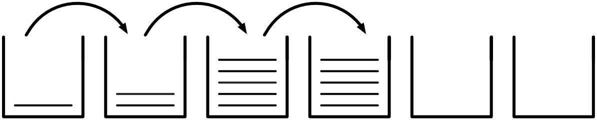




Beam-Search Decoding

- Score of an item $\langle s, q \rangle \circ x = \mathbf{w} \cdot \phi(\langle s, q \rangle, x)$





The Dependency Model

	[Clark et al., 2002]	C&C (dep)	Z&C	this work
Shift-Reduce	✗	✗	✓	✓
Dep. Model	✓	✓	✗	✓
Deriv. Feats	✗	✓	✓	✓

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence

The Dependency Model

	[Clark et al., 2002]	C&C (dep)	Z&C	this work
Shift-Reduce	✗	✗	✓	✓
Dep. Model	✓	✓	✗	✓
Deriv. Feats	✗	✓	✓	✓

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence
- The dependency model
  - given only gold depencey structures and exponentially many "correct" derivations hidden

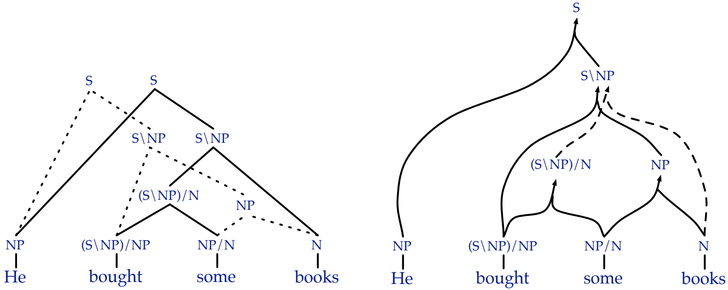
The Dependency Model

	[Clark et al., 2002]	C&C (dep)	Z&C	this work
Shift-Reduce	✗	✗	✓	✓
Dep. Model	✓	✓	✗	✓
Deriv. Feats	✗	✓	✓	✓

- The normal-form model [Zhang and Clark, 2011]
  - one unique gold derivation per sentence
- The dependency model
  - given only gold depencey structures and exponentially many "correct" derivations hidden
  - a method to query such an oracle
  - trained using the "early-update" variant of the violation-fixing perceptron [Huang et al., 2012]
  - similar to [Goldberg and Nivre 2012], our oracle to determine valid transition sequences

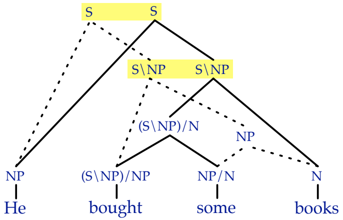
CCG Parse Forest

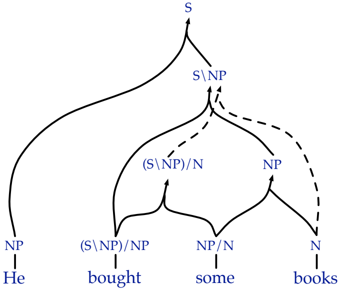
- We follow the definitions in [Clark and Curran, 2007; Miyao and Tsujji, 2002]
- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and *unfilled* dependencies
  - individual entries are *conjunctive* nodes and equivalence classes are *disjunctive* nodes



CCG Parse Forest

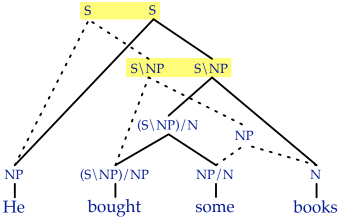
- We follow the definitions in [Clark and Curran, 2007; Miyao and Tsujji, 2002]
- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and *unfilled* dependencies
  - individual entries are *conjunctive* nodes and equivalence classes are *disjunctive* nodes





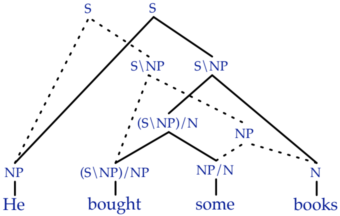
The Oracle Forest

- A **subset** of the **complete** forest
  - consistent with the gold-standard dependency structure
  - exponentially-sized and impossible to enumerate
- A dependency structure decomposes over derivations
  - dependencies are realized on conjunctive nodes
  - can count dependencies on-the-fly



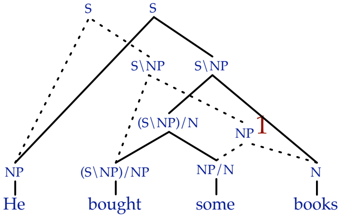
The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes



The Oracle Forest

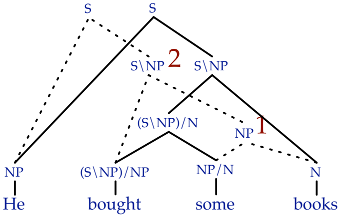
- intution 1: dependencies “live on” conjunctive nodes



$\langle$ *some*, NP/N_1, 1, *books* $\rangle$

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes

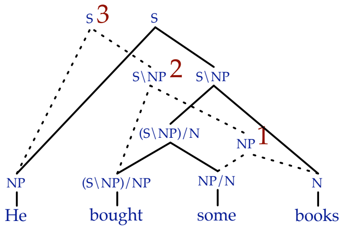


$\langle \textit{some}, NP/N_1, 1, \textit{books} \rangle$

$\langle \textit{bought}, (S \backslash NP_1)/NP_2, 2, \textit{books} \rangle$

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes



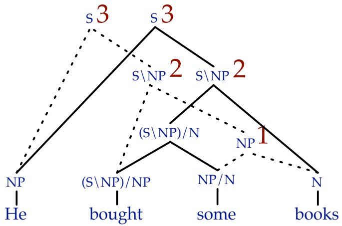
$\langle$*some*, NP/N_1, 1, *books*$\rangle$

$\langle$*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*$\rangle$

$\langle$*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*$\rangle$

The Oracle Forest

- intution 1: dependencies “live on” conjunctive nodes



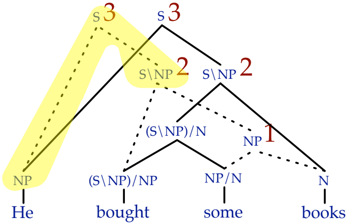
$\langle$*some*, NP/N_1, 1, *books*$\rangle$

$\langle$*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*$\rangle$

$\langle$*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*$\rangle$

The Oracle Forest

- intution 1: dependencies “live on” conjunctive nodes
- intution 2: a conj. node that has less than the max possible number of gold-standard dependencies is not gold (optimal substructure)



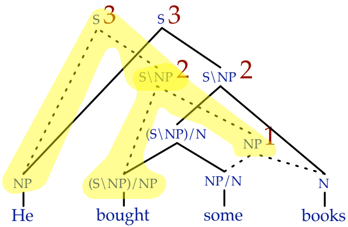
$\langle$ *some*, NP/N_1, 1, *books* $\rangle$

$\langle$ *bought*, $(S\backslash NP_1)/NP_2$, 2, *books* $\rangle$

$\langle$ *he*, $(S\backslash NP_1)/NP_2$, 1, *bought* $\rangle$

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes
- intution 2: a conj. node that has less than the max possible number of gold-standard dependencies is not gold (optimal substructure)



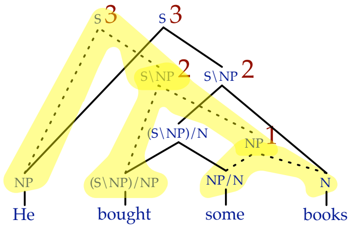
$\langle \textit{some}, \textit{NP}/\textit{N}_1, 1, \textit{books} \rangle$

$\langle \textit{bought}, (\textit{S}\backslash \textit{NP}_1)/\textit{NP}_2, 2, \textit{books} \rangle$

$\langle \textit{he}, (\textit{S}\backslash \textit{NP}_1)/\textit{NP}_2, 1, \textit{bought} \rangle$

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes
- intution 2: a conj. node that has less than the max possible number of gold-standard dependencies is not gold (optimal substructure)



$\langle$*some*, NP/N_1, 1, *books*$\rangle$

$\langle$*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*$\rangle$

$\langle$*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*$\rangle$

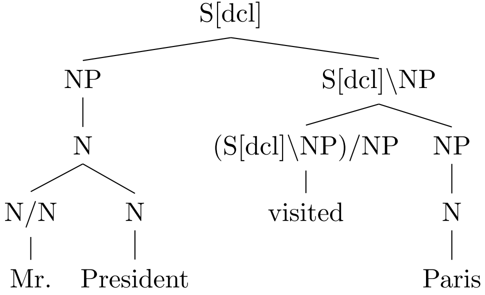
The Dependency Model Oracle

- The dependency oracle

$$f_d(\langle s, q\rangle, (x, c), \Phi_G) = \begin{cases} \mathit{true} & \text{if } s' \sim G \text{ or } s' \simeq G \\ \mathit{false} & \text{otherwise} \end{cases}$$

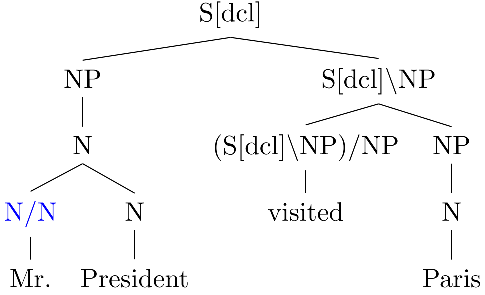
The Dependency Model Oracle

Canonical Shift-Reduce resembles bottom-up post-order traversal



The Dependency Model Oracle

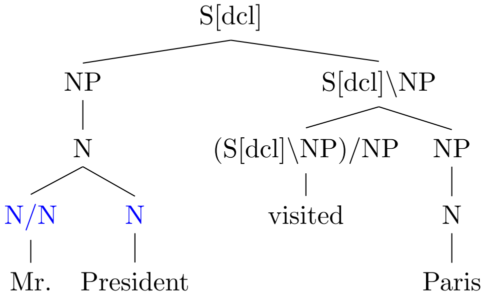
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift

The Dependency Model Oracle

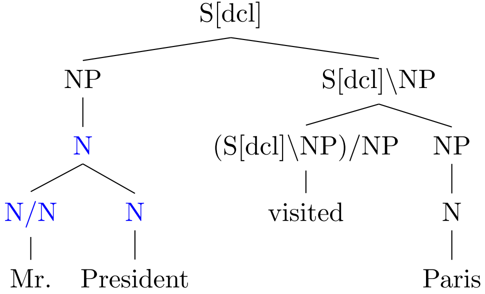
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift Shift

The Dependency Model Oracle

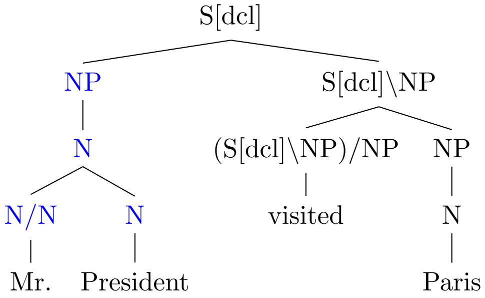
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift Shift Reduce

The Dependency Model Oracle

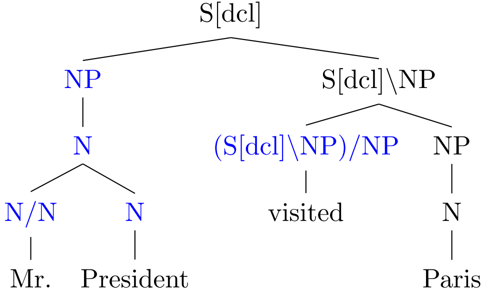
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift Shift Reduce Unary

The Dependency Model Oracle

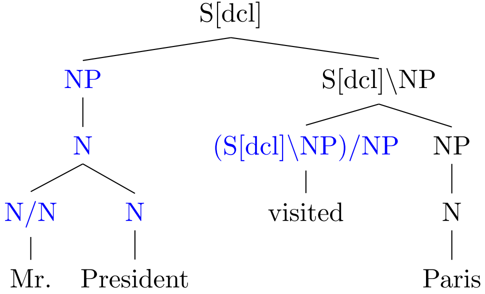
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift Shift Reduce Unary Shift

The Dependency Model Oracle

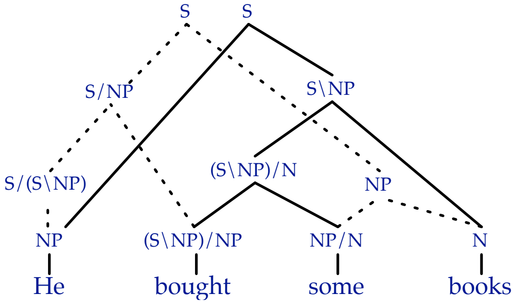
Canonical Shift-Reduce resembles bottom-up post-order traversal



Shift Shift Reduce Unary Shift Shift Unary Reduce Reduce

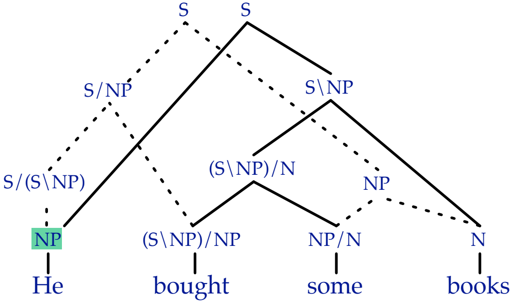
The Dependency Model Oracle

But this doesn't carry over to an oracle forest



The Dependency Model Oracle

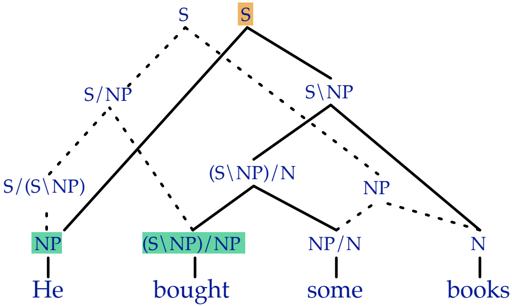
But this doesn't carry over to an oracle forest



Shift-*NP*

The Dependency Model Oracle

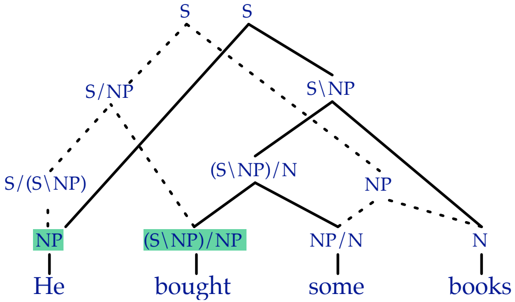
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$

The Dependency Model Oracle

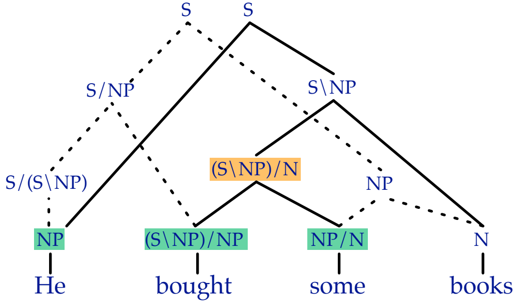
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$

The Dependency Model Oracle

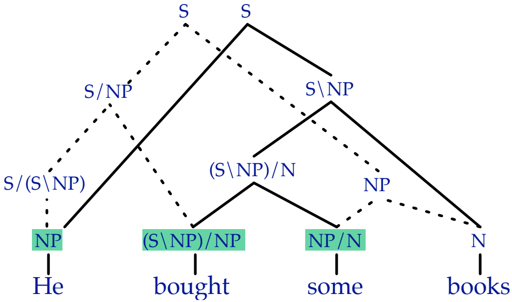
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$ Shift-NP/N

The Dependency Model Oracle

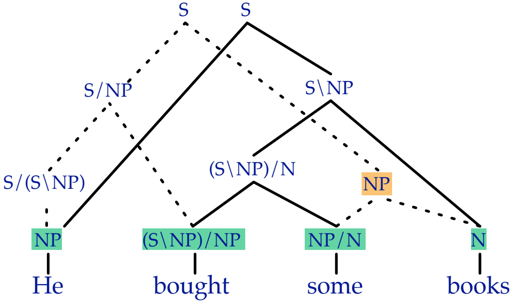
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$ Shift-NP/N

The Dependency Model Oracle

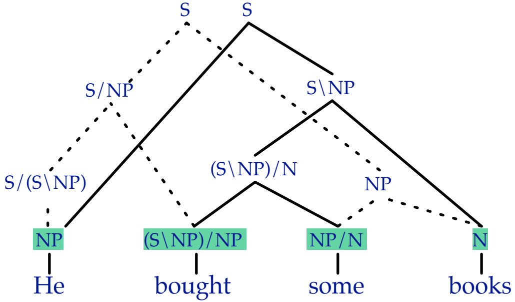
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$ Shift-NP/N Shift-N

The Dependency Model Oracle

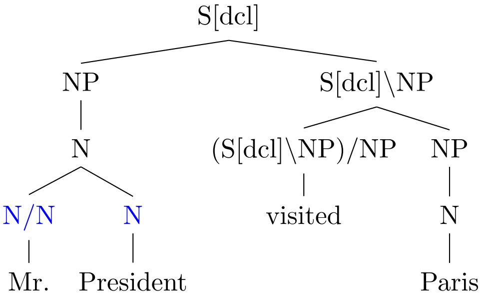
But this doesn't carry over to an oracle forest



Shift-NP Shift-$(S\backslash NP)/NP$ Shift-NP/N Shift-N

The Dependency Model Oracle

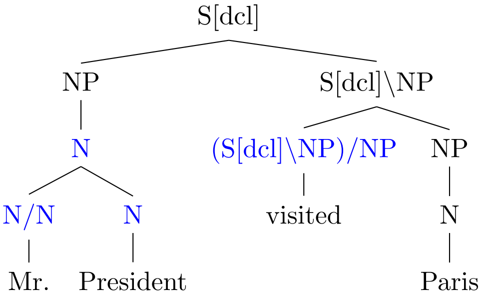
But this doesn't carry over to an oracle forest

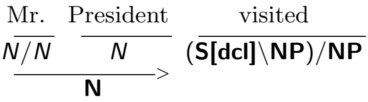


Mr.	President
N/N	**N**

The Dependency Model Oracle

But this doesn't carry over to an oracle forest





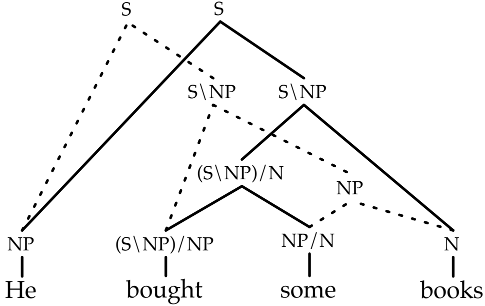
The Dependency Model Oracle

- The dependency oracle

$$f_d(\langle s, q \rangle, (x, c), \Phi_G) = \begin{cases} \textit{true} & \text{if } s' \sim G \text{ or } s' \simeq G \\ \textit{false} & \text{otherwise} \end{cases}$$

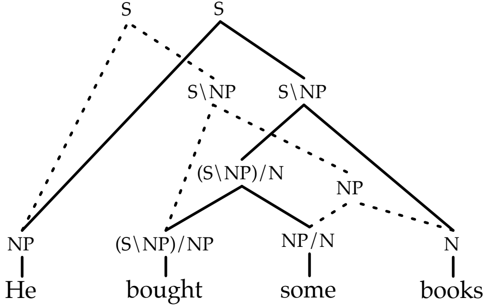
- Shared ancestor set
  - contains possible valid nodes an item should visit
  - is built on-the-fly during decoding for each action type
  - constructed with each *valid* action

The Dependency Model



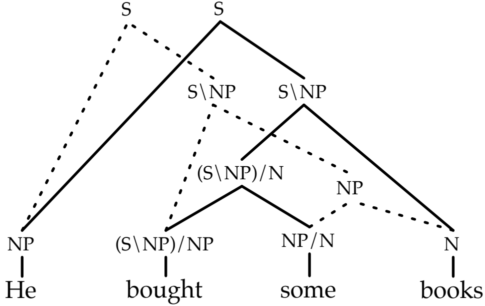
stack ($s_n, ..., s_1, s_0$)	$\mathcal{R}(c_{s_0})$

The Dependency Model



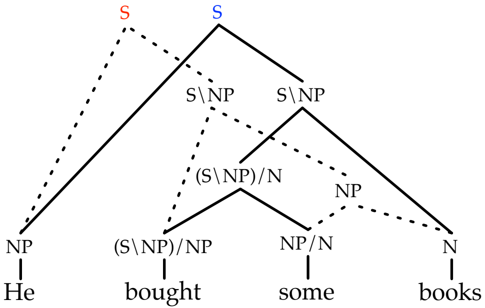
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	

The Dependency Model



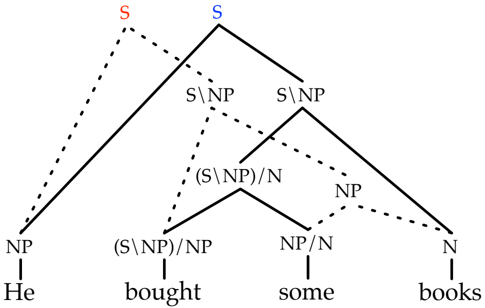
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	()

The Dependency Model



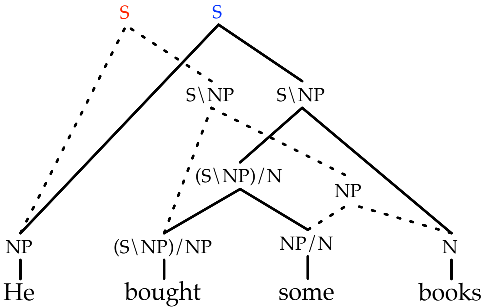
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	

The Dependency Model



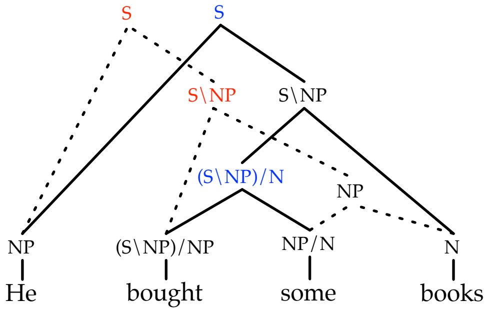
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	()
$NP\ (S\backslash NP)/NP$	$({\color{red}S}, {\color{blue}S})$

The Dependency Model



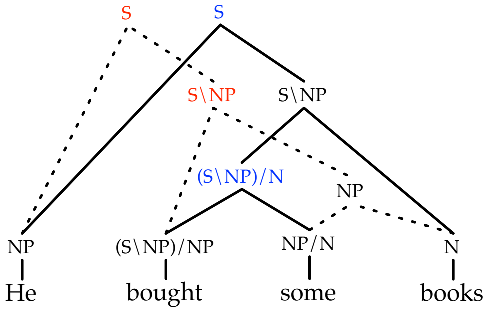
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	$({\color{red}S}, {\color{blue}S})$
$NP\ (S\backslash NP)/NP\ NP/N$	

The Dependency Model



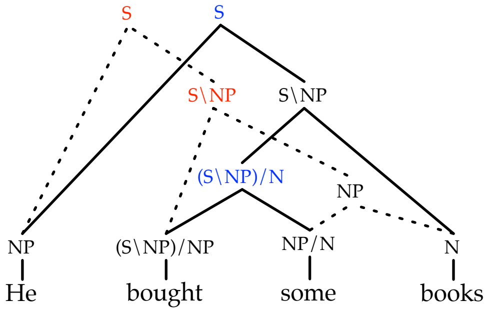
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	()
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	

The Dependency Model



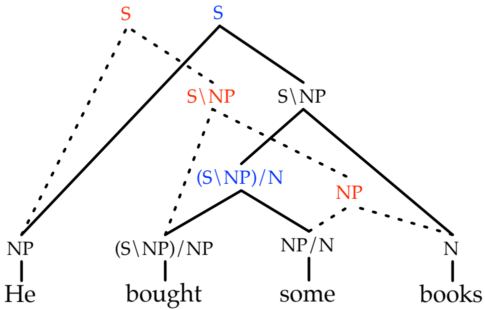
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$

The Dependency Model



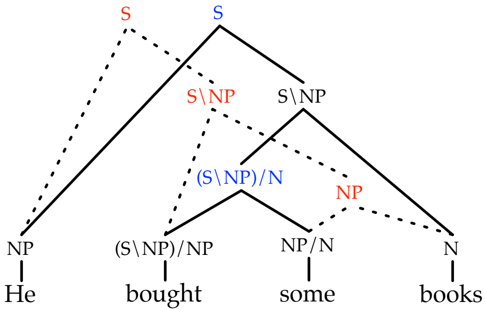
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	

The Dependency Model



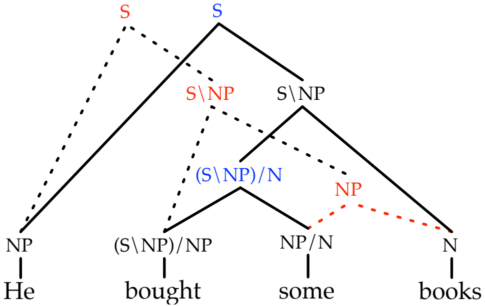
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	

The Dependency Model



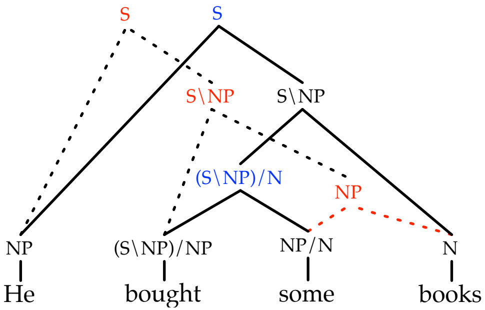
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)

The Dependency Model



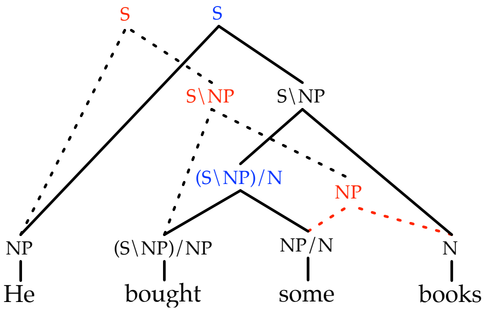
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	$({\color{red}S}, {\color{blue}S})$
$NP\ (S\backslash NP)/NP\ NP/N$	$({\color{red}S\backslash NP}, {\color{blue}(S\backslash NP)/N})$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	$({\color{red}NP})$
$NP\ (S\backslash NP)/NP\ NP$	

The Dependency Model



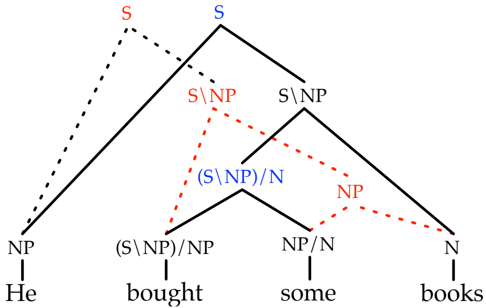
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$ ◀
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	

The Dependency Model



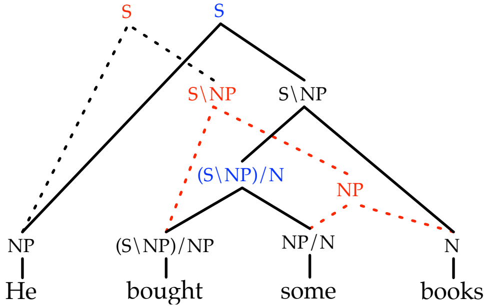
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$

The Dependency Model



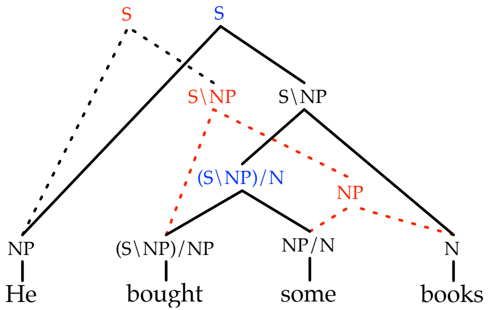
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$
$NP\ S\backslash NP$	

The Dependency Model



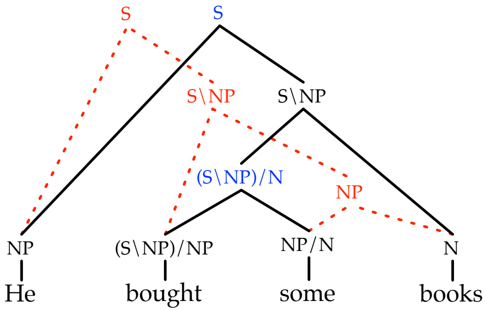
stack ($s_n, \ldots, s_1, s_0$)	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S) ◀
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$
$NP\ S\backslash NP$	

The Dependency Model



stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$
$NP\ S\backslash NP$	(S)

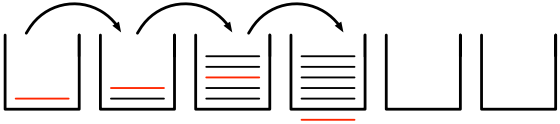
The Dependency Model



stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	$({\color{red}S}, {\color{blue}S})$
$NP\ (S\backslash NP)/NP\ NP/N$	$({\color{red}S\backslash NP}, {\color{blue}(S\backslash NP)/N})$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	$({\color{red}NP})$
$NP\ (S\backslash NP)/NP\ NP$	$({\color{red}S\backslash NP})$
$NP\ S\backslash NP$	$({\color{red}S})$
S	$()$

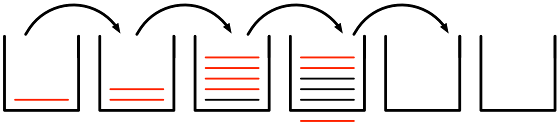
Online Training

- The normal-form model uses the perceptron with early update
  - only one correct sequence
  - “violation” is guaranteed [Huang et al., 2012]



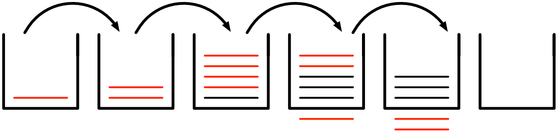
Online Training

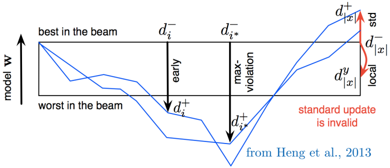
- Standard early update no longer valid for the dependency model
  - multiple correct items possible in each beam
  - "violation" is *not* guaranteed [Huang et al, 2012]



Online Training

- Standard early update no longer valid for the dependency model
  - multiple correct items possible in each beam
  - "violation" is *not* guaranteed [Huang et al, 2012]
  - $\mathbf{w} \leftarrow \mathbf{w} + \phi(\Pi_G[0]) - \phi(\mathcal{B}_i[0])$





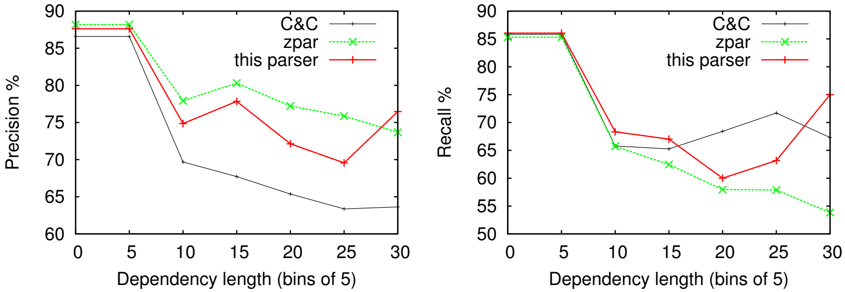
Experiments

- Standard split of CCGBank: training (2-21), dev (00) and test (23)
- Parser implemented as an extension of the C&C parser
  - unlike ZPAR, outputs dependencies directly
- Auto-pos for all experiments
- Supertagger prob. cutoff set to .0001 for both training and testing

Development Results

	LP	LR	LF	LSent. %	CatAcc. %	coverage %
this parser	86.29	**84.09**	**85.18**	**34.40**	92.75	100
Z&C	**87.15**	82.95	85.00	33.82	**92.77**	100
C&C (normal-form)	85.22	82.52	83.85	31.63	92.40	100
this parser	86.76	**84.90**	**85.82**	**34.72**	**93.20**	99.06 (C&C coverage)
Z&C	**87.55**	83.63	85.54	34.14	93.11	99.06 (C&C coverage)
C&C (hybrid)	–	–	85.25	–	–	99.06 (C&C coverage)
C&C (normal-form)	85.22	84.29	84.76	31.93	92.83	99.06 (C&C coverage)

Development Results



Development Results

category	LP % (t)	LP % (z)	LR % (t)	LR % (z)	LF % (t)	LF % (z)	freq.
$N/\mathbf{N}$	95.53	**95.77**	**95.83**	95.79	95.68	**95.78**	7288
$NP/\mathbf{N}$	96.53	**96.70**	**97.12**	96.59	**96.83**	96.65	4101
$(NP\backslash NP)/\mathbf{NP}$	81.64	**83.19**	**90.63**	89.24	85.90	**86.11**	2379
$(NP\backslash \mathbf{NP})/NP$	81.70	**82.53**	**88.91**	87.99	85.15	**85.17**	2174
$((S\backslash NP)\backslash(S\backslash NP))/\mathbf{NP}$	**77.64**	77.60	**72.97**	71.58	**75.24**	74.47	1147
$((S\backslash NP)\backslash(\mathbf{S}\backslash \mathbf{NP}))/NP$	75.78	**76.30**	**71.27**	70.60	**73.45**	73.34	1058
$((S[dcl]\backslash NP)/\mathbf{NP}$	83.94	**85.60**	**86.04**	84.30	**84.98**	84.95	917
$PP/\mathbf{NP}$	**77.06**	73.76	**73.63**	72.83	**75.31**	73.29	876
$((S[dcl]\backslash \mathbf{NP})/NP$	82.03	**85.32**	**83.26**	82.00	82.64	**83.63**	872
$((S\backslash NP)\backslash(\mathbf{S}\backslash \mathbf{NP}))$	**86.42**	84.44	86.19	**86.60**	**86.31**	85.51	746

Final Results

	LP %	LR %	LF %	LSent. %	CatAcc. %	coverage %
our parser	87.03	**85.08**	**86.04**	**35.69**	93.10	100
Z&C	**87.43**	83.61	85.48	35.19	**93.12**	100
C&C (normal-form)	85.58	82.85	84.20	32.90	92.84	100
our parser	87.04	**85.16**	**86.09**	**35.84**	93.13	99.58 (C&C coverage)
Z&C	**87.43**	83.71	85.53	35.34	**93.15**	99.58 (C&C coverage)
C&C (hybrid)	86.17	84.74	85.45	32.92	92.98	99.58 (C&C coverage)
C&C (normal-form)	85.48	84.60	85.04	33.08	92.86	99.58 (C&C coverage)

$+0.5$ over Z&C, and $+0.6$ and $+1.5$ over C&C hybrid and normal-form models, resp.

Conclusions

- Introduced the first dependency model for shift-reduce CCG parsing
  - only gold-standard dependencies are needed for training
  - the oracle encodes *exponentially* many derivations
  - achieved the best accuracy for shift-reduce CCG parsing

The End

Thank You!