Structured Learning with Inexact Search: Advances in Shift-Reduce CCG Parsing

This work is made possible and fully supported by the Carnegie Trust for the Universities of Scotland and the Cambridge Trust.

Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

- Decomposition: $\mathcal{D}(y)$

Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

- Decomposition: $\mathcal{D}(y)$
- Scoring: $score(\Phi(d))$

Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

- Decomposition: $\mathcal{D}(y)$
- Scoring: $score(\Phi(d))$
- Summing: $\sum_{d \in \mathcal{D}(y)}$

Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

- Decomposition: $\mathcal{D}(y)$
- Scoring: $\mathit{score}(\Phi(d))$
- Summing: $\sum_{d \in \mathcal{D}(y)}$
- Search: $\arg\max_{y \in \mathcal{Y}_x}$

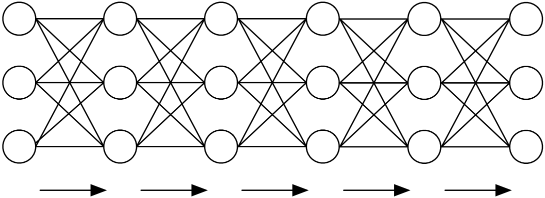
Structured Prediction in NLP

$$y^* = \arg\max_{y \in \mathcal{Y}_x} \sum_{d \in \mathcal{D}(y)} \text{score}(\Phi(d))$$

- Decomposition: $\mathcal{D}(y)$
- Scoring: $score(\Phi(d))$
- Summing: $\sum_{d \in \mathcal{D}(y)}$
- Search: $\arg\max_{y \in \mathcal{Y}_x}$

$\mathcal{Y}_x$ is exponentially-sized and prohibitive to enumerate.

Structured Prediction: Sequence Labelling



MEMM [McCallum et al., 2000]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$

MEMM [McCallum et al., 2000]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$

$$= \prod_{i=1}^{m} p(y_i | y_1 \ldots, y_{i-1}, x_1, \ldots, x_m)$$

MEMM [McCallum et al., 2000]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$

$$= \prod_{i=1}^{m} p(y_i | y_1 \ldots, y_{i-1}, x_1, \ldots, x_m)$$

$$= \prod_{i=1}^{m} p(y_i | y_{i-1}, x_1, \ldots, x_m)$$

MEMM [McCallum et al., 2000]

$$
\begin{aligned}
& p(y_1, \ldots, y_m | x_1, \ldots, x_m) \\
&= \prod_{i=1}^{m} p(y_i | y_1 \ldots, y_{i-1}, x_1, \ldots, x_m) \\
&= \prod_{i=1}^{m} p(y_i | y_{i-1}, x_1, \ldots, x_m) \\
&= \prod_{i=1}^{m} \frac{\exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i)\}}{\sum_{y_i'} \exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i')\}}
\end{aligned}
$$

CRF [Lafferty et al., 2001]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$

CRF [Lafferty et al., 2001]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$

$$= \frac{1}{z} \exp\{\sum_{i=1}^{m} \sum_{j=1}^{F} \omega_j \phi_j(y_{i-1}, y_i, x, i)\}$$

CRF [Lafferty et al., 2001]

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m)$$
$$= \frac{1}{z} \exp\{\sum_{i=1}^{m} \sum_{j=1}^{F} \omega_j \phi_j(y_{i-1}, y_i, x, i)\}$$

$$z = \sum_{y_{1:m} \in \mathcal{Y}_x} \exp\{\sum_{i=1}^{m} \sum_{j=1}^{F} \omega_j \phi_j(y_{i-1}, y_i, x, i)\}$$

MEMM and CRF

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m) = \prod_{i=1}^{m} \frac{\exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i)\}}{\sum_{y_i'} \exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i')\}}$$

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m) = \frac{1}{z} \exp\{\sum_{i=1}^{m} \sum_{j=1}^{F} \omega_j \phi_j(y_{i-1}, y_i, x, i)\}$$

- Feature function: Φ
- Structured output: y
- Search: dynamic programming + Viterbi decoding
- $\arg\max_{y_{1:m}} p(y_1, \ldots, y_m | x_1, \ldots, x_m)$

MEMM and CRF

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m) = \prod_{i=1}^{m} \frac{\exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i)\}}{\sum_{y_i'} \exp\{\mathbf{w} \cdot \Phi(x_1, \ldots, x_m, i, y_{i-1}, y_i')\}}$$

$$p(y_1, \ldots, y_m | x_1, \ldots, x_m) = \frac{1}{z} \exp\{\sum_{i=1}^{m} \sum_{j=1}^{F} \omega_j \phi_j(y_{i-1}, y_i, x, i)\}$$

- Feature function: Φ
- Structured output: y
- Search: dynamic programming + Viterbi decoding
- $\arg\max_{y_{1:m}} p(y_1, \ldots, y_m | x_1, \ldots, x_m)$

The Structured Perceptron [Collins, 2002]

```
w ← 0                                            ▷ the input is the training set {(x_i, y_i)}_{i=1}^n
while not converged do
  for i ← 1, ..., n do
    y* ← arg max_{y∈GEN(x_i)} w · Φ(x_i, y)           ▷ obtain model prediction
    if y* ≠ y_i then                                 ▷ y* not correct
      w ← w + Φ(x_i, y_i) − Φ(x_i, y*)               ▷ online update
```

The Structured Perceptron [Collins, 2002]

$\mathbf{w} \leftarrow \mathbf{0}$ $\triangleright$ the input is the training set $\{(x_i, y_i)\}_{i=1}^{n}$
while not converged **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $y^* \leftarrow \arg\max_{y \in \mathrm{GEN}(x_i)} \mathbf{w} \cdot \Phi(x_i, y)$ $\triangleright$ obtain model prediction
    if $y^* \neq y_i$ **then** $\triangleright$ y^* not correct
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, y^*)$ $\triangleright$ online update

The Structured Perceptron [Collins, 2002]

$\mathbf{w} \leftarrow \mathbf{0}$ ▷ the input is the training set $\{(x_i, y_i)\}_{i=1}^n$
while not converged **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $y^* \leftarrow \underset{y \in \mathrm{GEN}(x_i)}{\arg\max}\ \mathbf{w} \cdot \Phi(x_i, y)$ ▷ obtain model prediction
    if $y^* \neq y_i$ **then** ▷ y^* not correct
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, y^*)$ ▷ online update

- Feature function: Φ
- Structured output: y
- Search: dynamic programming

The Structured Perceptron [Collins, 2002]

$\mathbf{w} \leftarrow \mathbf{0}$ ▷ the input is the training set $\{(x_i, y_i)\}_{i=1}^{n}$
while not converged **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $y^* \leftarrow \arg\max_{y \in \text{GEN}(x_i)} \mathbf{w} \cdot \Phi(x_i, y)$ ▷ obtain model prediction
    if $y^* \neq y_i$ **then** ▷ y^* not correct
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, y^*)$ ▷ online update

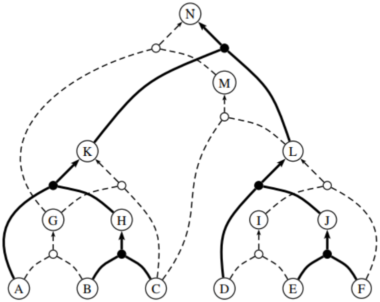
- Feature function: Φ
- Structured output: y
- Search: dynamic programming
  - beam search (the incremental structured perceptron [Collins and Roark, 2004])

The Structured Perceptron [Collins, 2002]

$\mathbf{w} \leftarrow \mathbf{0}$ ▷ the input is the training set $\{(x_i, y_i)\}_{i=1}^{n}$
while not converged **do**
  for $i \leftarrow 1, \ldots, n$ **do**
    $y^* \leftarrow \underset{y \in \mathrm{GEN}(x_i)}{\arg\max}\ \mathbf{w} \cdot \Phi(x_i, y)$ ▷ obtain model prediction
    if $y^* \neq y_i$ **then** ▷ y^* not correct
      $\mathbf{w} \leftarrow \mathbf{w} + \Phi(x_i, y_i) - \Phi(x_i, y^*)$ ▷ online update

- Feature function: Φ
- Structured output: y
- Search: dynamic programming
  - beam search (the incremental structured perceptron [Collins and Roark, 2004])
  - dynamic programming + cube pruning [Chiang, 2007]

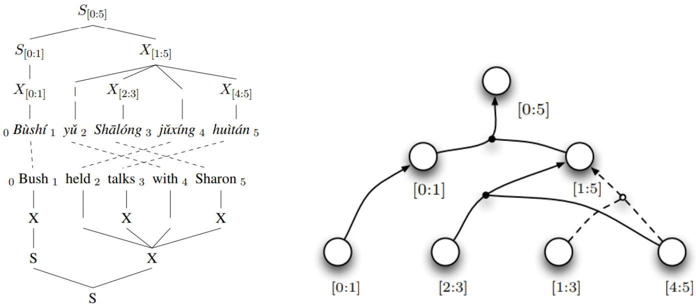
Structured Perceptron with Inexact Search [Huang et al., 2012]



Graph-based dependency parsing

[Zhang and McDonald, 2012; Zhang et al., 2013]

Structured Perceptron with Inexact Search [Huang et al., 2012]



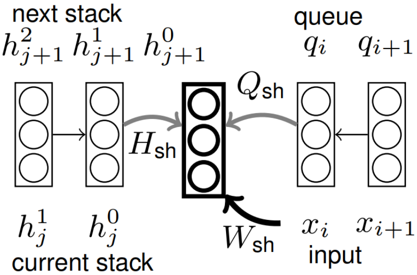
Hierarchical phrase-based translation

[Zhao et al., 2014]

Neural Network Models

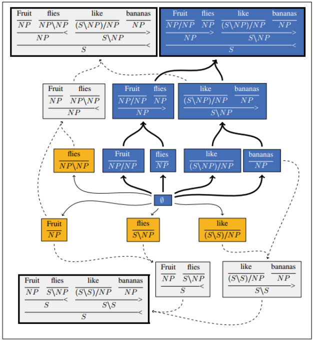
- Sequence-to-Sequence [Sutskever et al., 2014]
  - training: per-step cross-entropy
  - test: $p(y_1, \ldots, y_n | x_1, \ldots, x_m) = \prod_{t=1}^{n} p(y_t | y_1 \ldots, y_{t-1}, \mathbf{c})$
  - search: $y^* = \arg\max_{y \in \mathcal{Y}_x} p(y|x)$
- Representation learning: RNN, LSTM, CNN [Gehring et al., 2017]
- Search: greedy, beam search (no search at training time)
- Structured learning: [Ranzato et al., 2016; Wiseman and Rush, 2016]
- most recent: [Edunov et al., 2017]

Neural Network Models + Structured Perceptron-Inspired Updates



Watanabe and Sumita, 2015 uses a variant of Max Violation.

Neural Network Models + Structured Perceptron-Inspired Updates



Lee et al., 2016 extends Max Violation to All Violation.

Outline

- Three models for shift-reduce CCG parsing
  - representation learning:
  - structured learning:
  - search:

Outline

- Three models for shift-reduce CCG parsing
  - representation learning: struct. perceptron, Elman RNN, and LSTM
  - structured learning:
  - search:

Outline

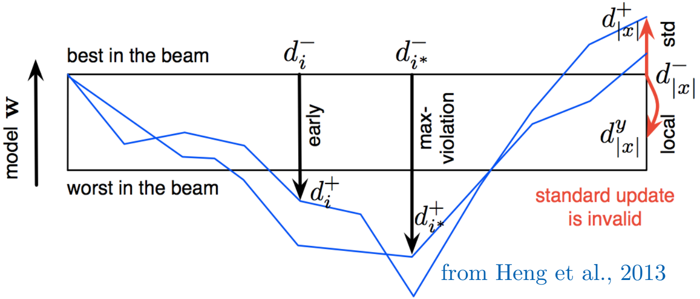
- Three models for shift-reduce CCG parsing
  - representation learning: struct. perceptron, Elman RNN, and LSTM
  - structured learning: sequence-level training (global vs. local)
  - search:

Outline

- Three models for shift-reduce CCG parsing
  - representation learning: struct. perceptron, Elman RNN, and LSTM
  - structured learning: sequence-level training (global vs. local)
  - search: beam search for both training and testing

Outline

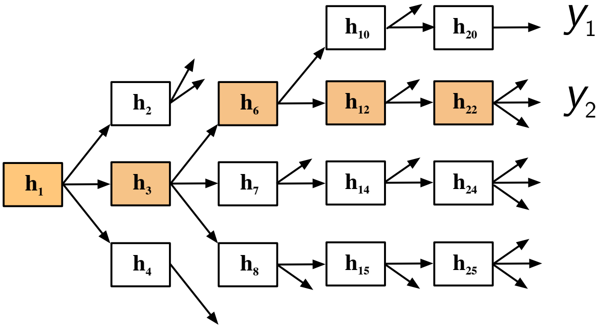
- Three models for shift-reduce CCG parsing
  - representation learning: struct. perceptron, Elman RNN, and LSTM
  - structured learning: sequence-level training (global vs. local)
  - search: beam search for both training and testing



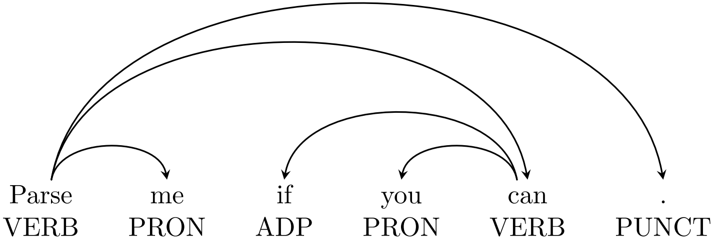
from Heng et al., 2013

Outline

- Three models for shift-reduce CCG parsing
  - representation learning: struct. perceptron, Elman RNN, and LSTM
  - structured learning: sequence-level training (global vs. local)
  - search: beam search for both training and testing

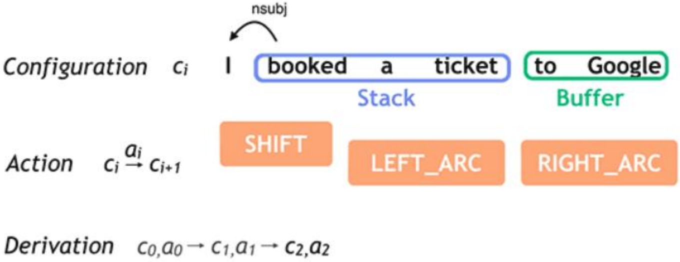


Dependency Parsing



Google SyntaxNet output

Transition-based Dependency Parsing



source: Google SyntaxNet

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

the books which John likes

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

$$\frac{\text{the}}{NP/N} \quad \frac{\text{books}}{N} \quad \frac{\text{which}}{(NP\backslash NP)/(S/NP)} \quad \frac{\text{John}}{NP} \quad \frac{\text{likes}}{(S\backslash NP)/NP}$$

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

$$\dfrac{\text{the}}{NP/N} \quad \dfrac{\text{books}}{N} \quad \dfrac{\text{which}}{(NP\backslash NP)/(S/NP)} \quad \dfrac{\text{John}}{NP} \quad \dfrac{\text{likes}}{(S\backslash NP)/NP}$$

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

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

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

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

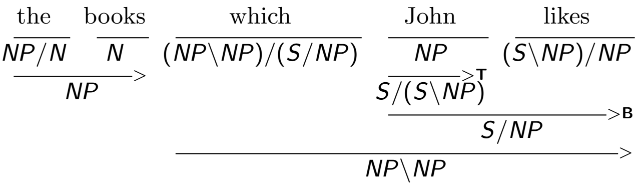
Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

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

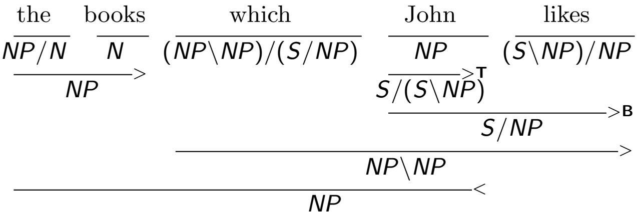
Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)



Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)



Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

- Parsing CCG
  - Supertagging (regular language; 1000 tags vs. 50 for CFG)
  - Parsing (mildly context-sensitive; only a dozen rules vs. 500K for CFG [Petrov and Klein, 2007])

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

- Parsing CCG
  - Supertagging (regular language; 1000 tags vs. 50 for CFG)
  - Parsing (mildly context-sensitive; only a dozen rules vs. 500K for CFG [Petrov and Klein, 2007])

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)
- Parsing CCG – structured learning
  - Supertagging (regular language; 1000 tags vs. 50 for CFG)
  - Parsing (mildly context-sensitive; only a dozen rules vs. 500K for CFG [Petrov and Klein, 2007])

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)

- Parsing CCG – structured learning
  - Supertagging (regular language; 1000 tags vs. 50 for CFG)
  - Parsing (mildly context-sensitive; only a dozen rules vs. 500K for CFG [Petrov and Klein, 2007])

- Dual Decomposition, Belief Propogation [Auli and Lopez, 2011]

Shift-Reduce CCG Parsing

- Combinatory Categorial Grammar (CCG)
- Parsing CCG – structured learning
  - Supertagging (regular language; 1000 tags vs. 50 for CFG)
  - Parsing (mildly context-sensitive; only a dozen rules vs. 500K for CFG [Petrov and Klein, 2007])
- Dual Decomposition, Belief Propogation [Auli and Lopez, 2011]
- Remains to be the most competitive formalism for recovering "deep" dependencies (from coordination, control, extraction etc.) [Rimell et al., 2009; Nivre et al., 2010]

Shift-Reduce CCG Parsing

the books which John likes

Shift-Reduce CCG Parsing

the	books	which	John	likes
NP/N				

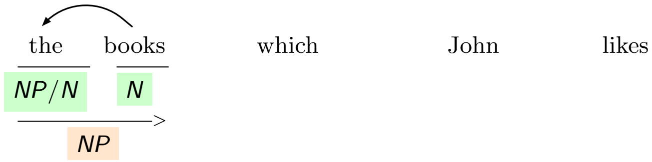
SH

Shift-Reduce CCG Parsing

the	books	which	John	likes
NP/N	N			

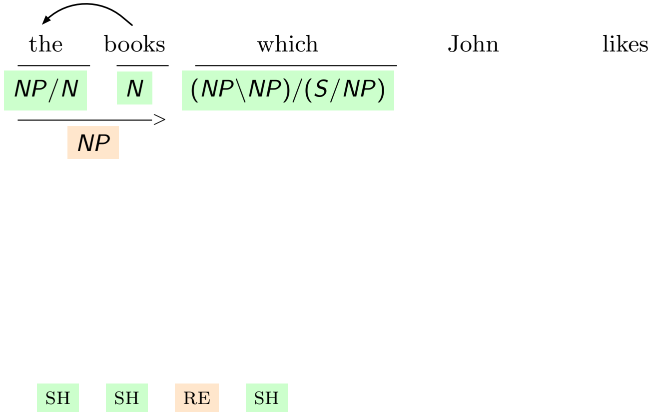
SH SH

Shift-Reduce CCG Parsing

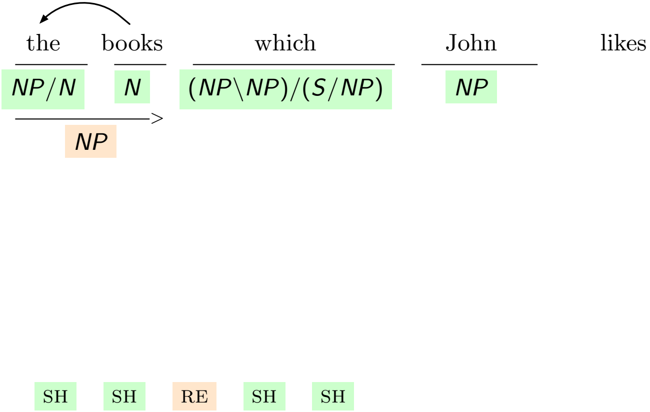


SH SH RE

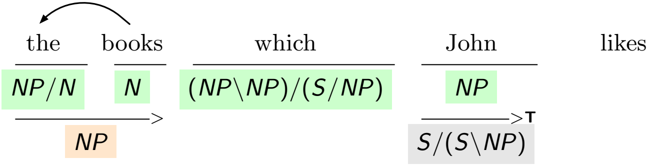
Shift-Reduce CCG Parsing



Shift-Reduce CCG Parsing

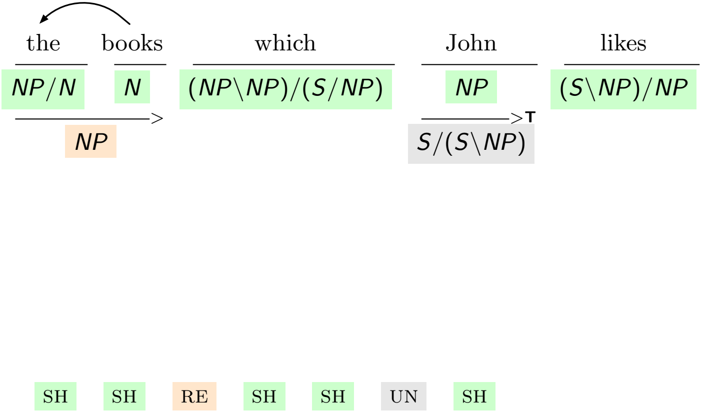


Shift-Reduce CCG Parsing

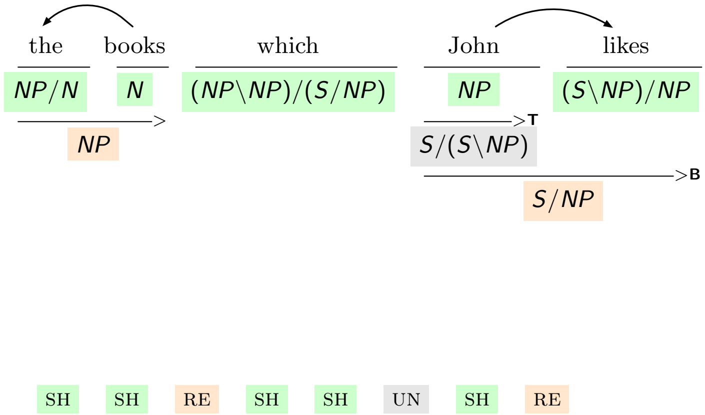


SH SH RE SH SH UN

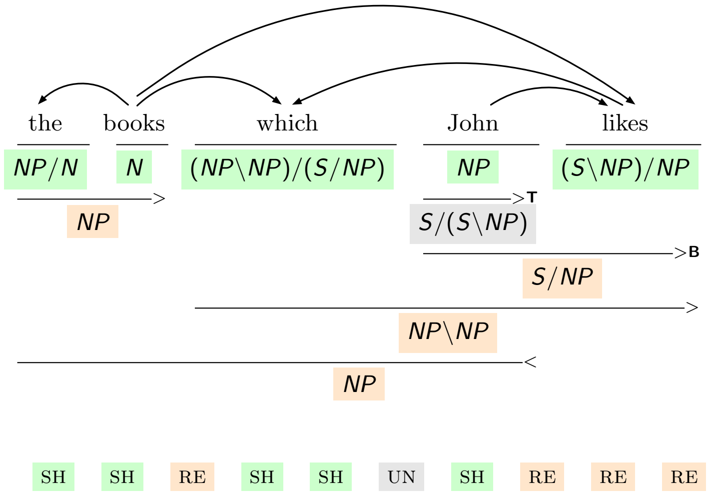
Shift-Reduce CCG Parsing



Shift-Reduce CCG Parsing



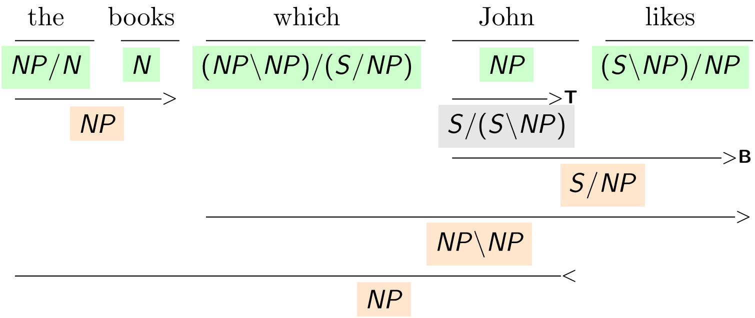
Shift-Reduce CCG Parsing



Model 1

[Xu et al., ACL 2014]

Standard Training: Greedy Local Model



SH SH RE SH SH UN SH RE RE RE

Standard Training: Greedy Local Model

- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

Standard Training: Greedy Local Model

- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

Standard Training: Greedy Local Model

- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

Standard Training: Greedy Local Model

- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

Standard Training: Greedy Local Model

- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

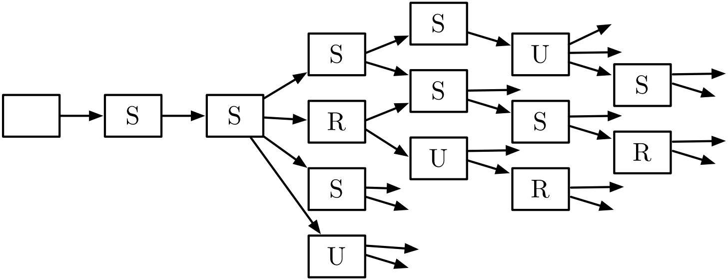
Standard Training: Greedy Local Model

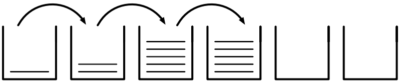
- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$
- No search at training time, can use beam search decoding

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

Global Structured Training
[Collins and Roark, 2004]

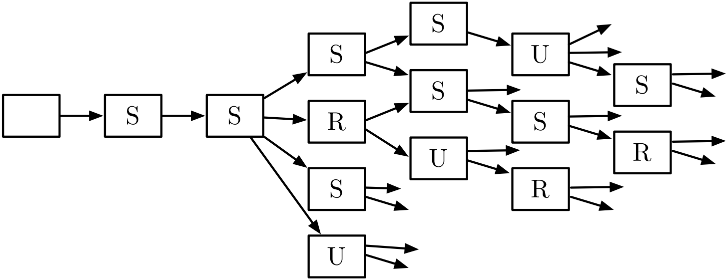
- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$

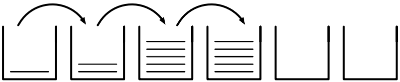




Global Structured Training
[Collins and Roark, 2004]

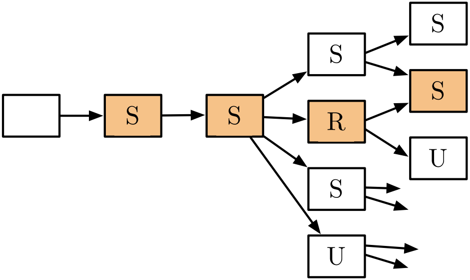
- Score of an action $a = \mathbf{w} \cdot \phi(\langle s, q \rangle, a)$

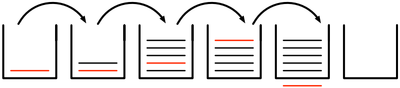




Global Structured Training
[Collins and Roark, 2004]

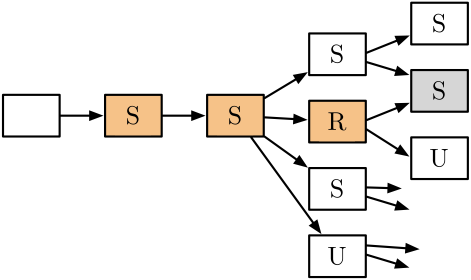
- Structured perceptron update: $\mathbf{w} \leftarrow \mathbf{w} + \phi(x_i, y_{ij}) - \phi(x_i, \mathcal{B}_j[0])$

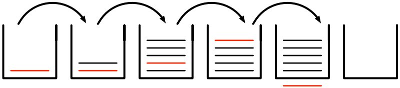




Global Structured Training
[Collins and Roark, 2004]

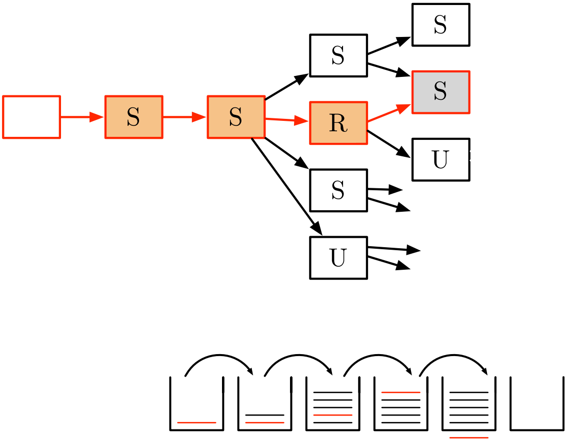
- Structured perceptron update: $\mathbf{w} \leftarrow \mathbf{w} + \phi(x_i, y_{ij}) - \phi(x_i, \mathcal{B}_j[0])$





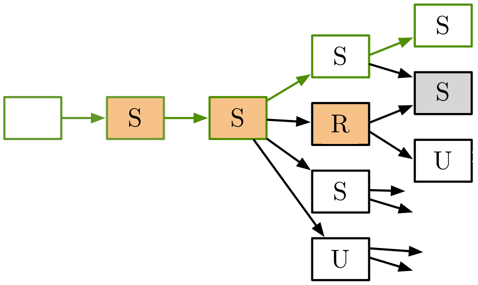
Global Structured Training
[Collins and Roark, 2004]

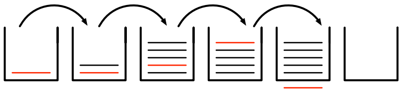
- Structured perceptron update: $\mathbf{w} \leftarrow \mathbf{w} + \phi(x_i, y_{ij}) - \phi(x_i, \mathcal{B}_j[0])$



Global Structured Training
[Collins and Roark, 2004]

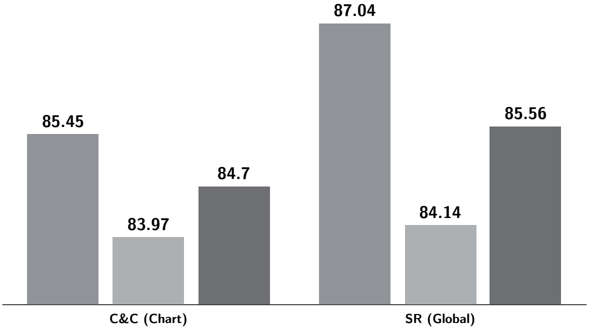
- Structured perceptron update: $\mathbf{w} \leftarrow \mathbf{w} + \phi(x_i, y_{ij}) - \phi(x_i, \mathcal{B}_j[0])$



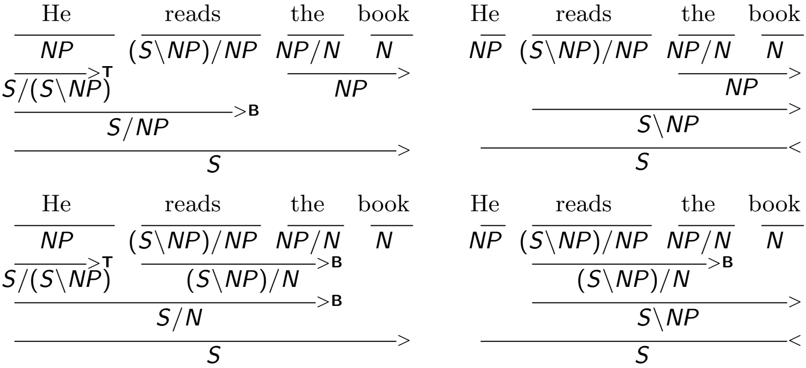


Global Structured Training for CCG
[Zhang and Clark, 2011]

- Conditional log-linear vs. linear
- Dynamic programming vs. beam search



Spurious Ambiguity in CCG



$\langle$*the*, *book*$\rangle$
$\langle$*reads*, *book*$\rangle$
$\langle$*reads*, *he*$\rangle$

In general, exponentially many!

Motivation: Dependency Model

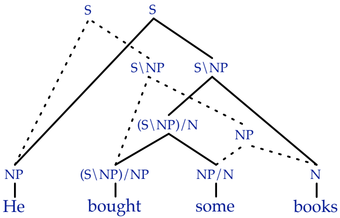
- The derivation is just a “trace” of the semantic interpretation [Steedman, 2000]

Motivation: Dependency Model

- The derivation is just a “trace” of the semantic interpretation [Steedman, 2000]
  - an elegant solution to the spurious ambiguity problem
  - gold-standard data cheaper to obtain
  - optimizing for evaluation

Model 1: The Dependency Model

- Use dependencies as the ground truth
  - encoding exponentially many “correct” paths



Model 1: The Dependency Model

- Use dependencies as the ground truth
  - encoding exponentially many “correct” paths
  - path selection is a hidden variable

Model 1: The Dependency Model

- Use dependencies as the ground truth
  - encoding exponentially many "correct" paths
  - path selection is a hidden variable
- A dependency oracle algorithm – online hypergraph search

Model 1: The Dependency Model

- Use dependencies as the ground truth
  - encoding exponentially many “correct” paths
  - path selection is a hidden variable
- A dependency oracle algorithm – online hypergraph search
- A learning algorithm adapting early update (under the violation-fixing struct. perceptron [Huang et al., 2012])

Model 1: The Dependency Model

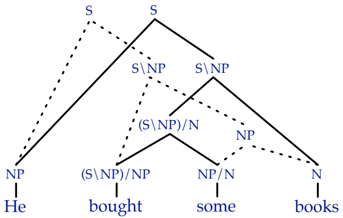
- Use dependencies as the ground truth
  - encoding exponentially many "correct" paths
  - path selection is a hidden variable
- A dependency oracle algorithm – online hypergraph search
- A learning algorithm adapting early update (under the violation-fixing struct. perceptron [Huang et al., 2012])
- Beam search – global structured learning

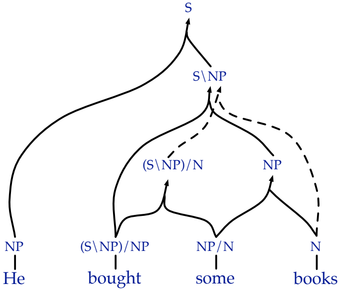
The Dependency Model

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

CCG Parse Forest

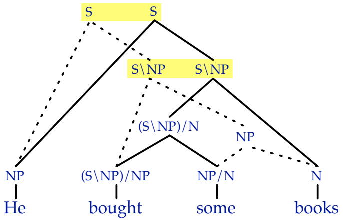
- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and *unfilled* dependencies
  - individual entries are *conjunctive* nodes and equivalence classes are *disjunctive* nodes

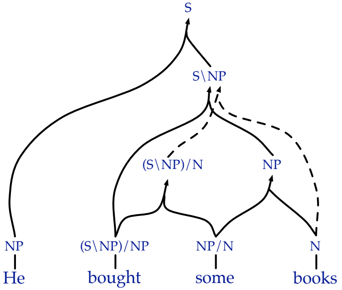




CCG Parse Forest

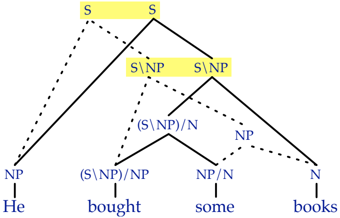
- Compactly represents all derivation and dependency structure pair
- Grouping together equivalent chart entries
  - identical category, head and *unfilled* dependencies
  - individual entries are *conjunctive* nodes and equivalence classes are *disjunctive* nodes





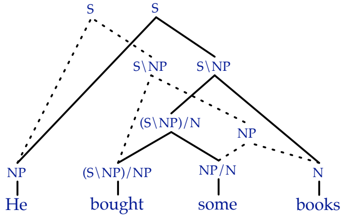
The Oracle Forest

- 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



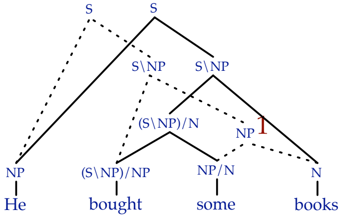
The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes



The Oracle Forest

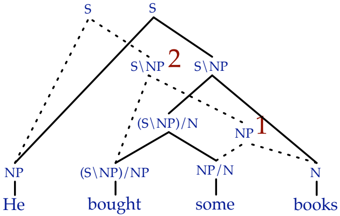
- intution 1: dependencies "live on" conjunctive nodes



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

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes

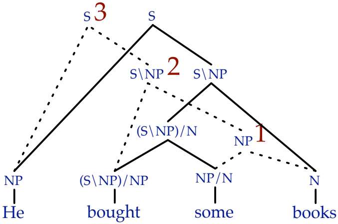


$\langle some, NP/N_1, 1, books \rangle$

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

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes



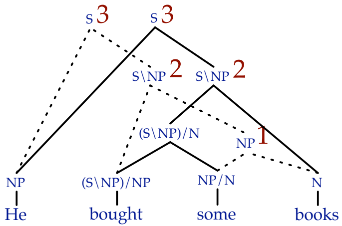
⟨*some*, NP/N_1, 1, *books*⟩

⟨*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*⟩

⟨*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*⟩

The Oracle Forest

- intution 1: dependencies "live on" conjunctive nodes



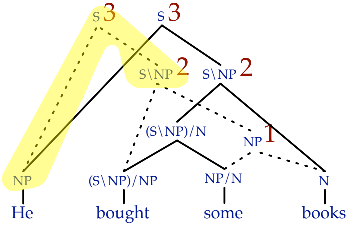
⟨*some*, NP/N_1, 1, *books*⟩

⟨*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*⟩

⟨*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*⟩

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)



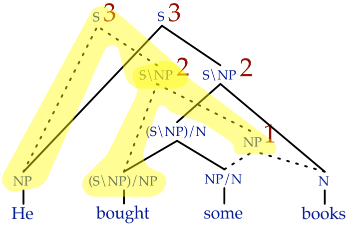
⟨*some*, NP/N_1, 1, *books*⟩

⟨*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*⟩

⟨*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*⟩

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)



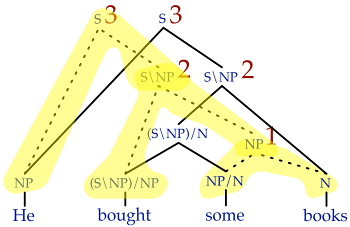
⟨*some*, NP/N_1, 1, *books*⟩

⟨*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*⟩

⟨*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*⟩

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)



⟨*some*, NP/N_1, 1, *books*⟩

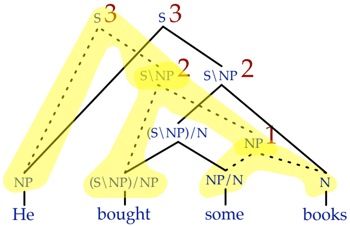
⟨*bought*, $(S\backslash NP_1)/NP_2$, 2, *books*⟩

⟨*he*, $(S\backslash NP_1)/NP_2$, 1, *bought*⟩

Shift-Reduce Dependency Oracle

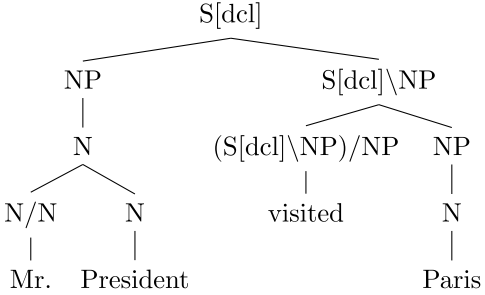
- The dependency oracle

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



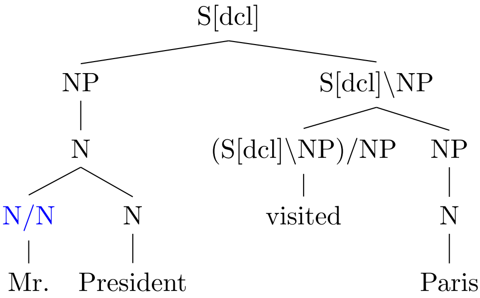
The Dependency Model Oracle

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



The Dependency Model Oracle

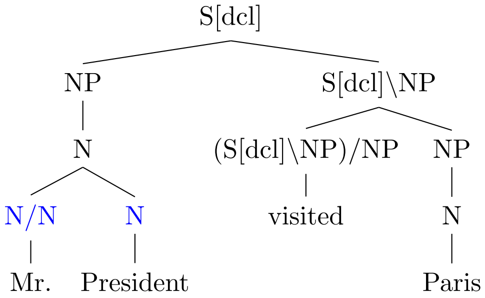
Canonical Shift-Reduce is bottom-up post-order traversal



Shift

The Dependency Model Oracle

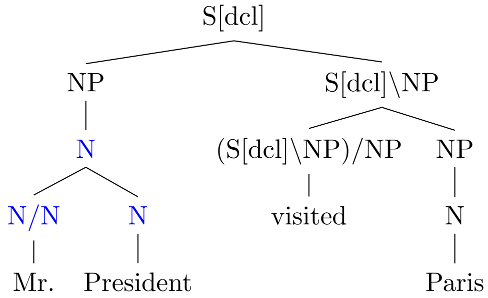
Canonical Shift-Reduce is bottom-up post-order traversal



Shift Shift

The Dependency Model Oracle

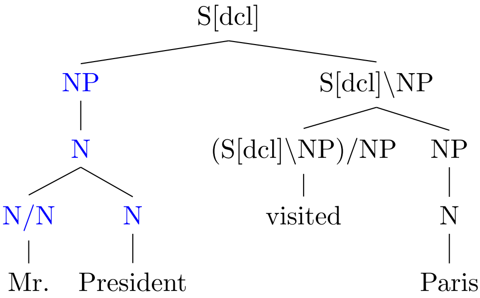
Canonical Shift-Reduce is bottom-up post-order traversal



Shift Shift Reduce

The Dependency Model Oracle

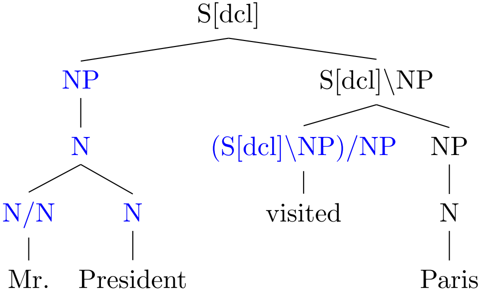
Canonical Shift-Reduce is bottom-up post-order traversal



Shift Shift Reduce Unary

The Dependency Model Oracle

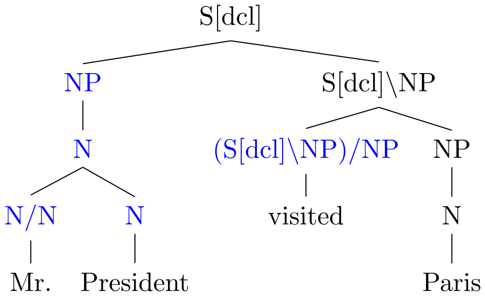
Canonical Shift-Reduce is bottom-up post-order traversal



Shift Shift Reduce Unary Shift

The Dependency Model Oracle

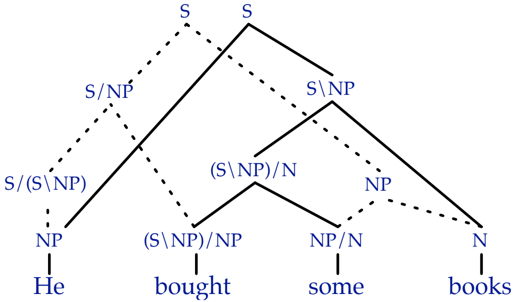
Canonical Shift-Reduce is bottom-up post-order traversal



Shift Shift Reduce Unary Shift Shift Unary Reduce Reduce

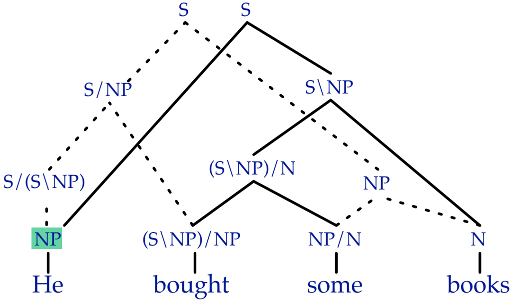
The Dependency Model Oracle

But this doesn't carry over to an oracle forest



The Dependency Model Oracle

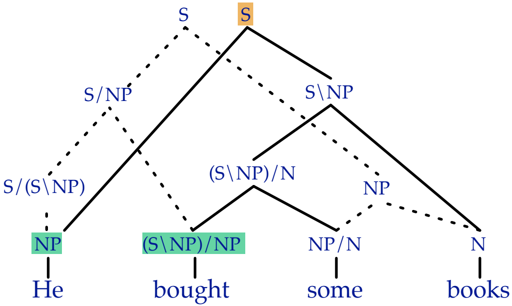
But this doesn't carry over to an oracle forest



Shift-*NP*

The Dependency Model Oracle

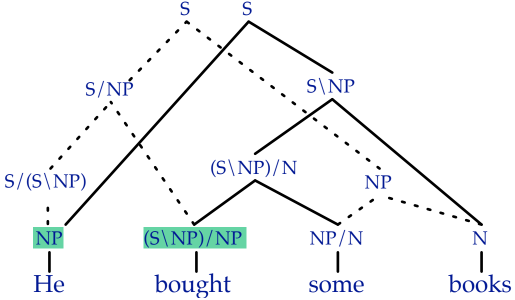
But this doesn't carry over to an oracle forest



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

The Dependency Model Oracle

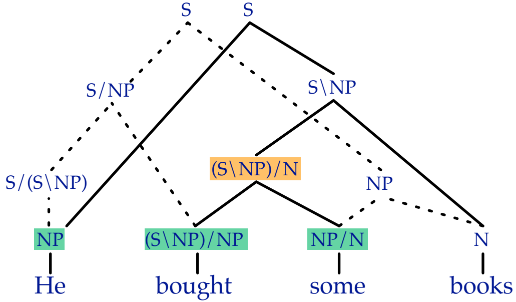
But this doesn't carry over to an oracle forest



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

The Dependency Model Oracle

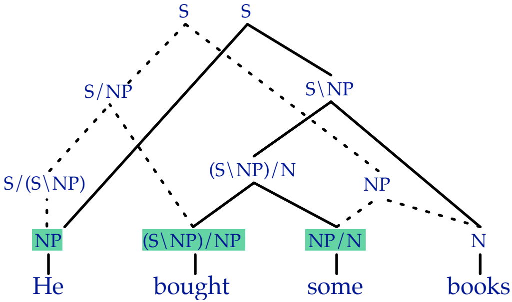
But this doesn't carry over to an oracle forest



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

The Dependency Model Oracle

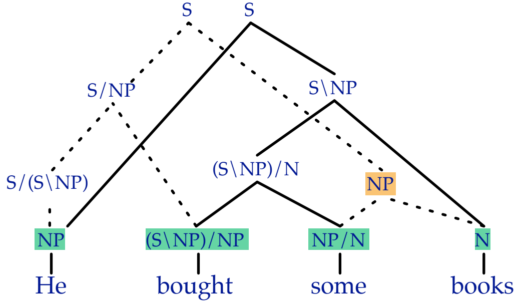
But this doesn't carry over to an oracle forest



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

The Dependency Model Oracle

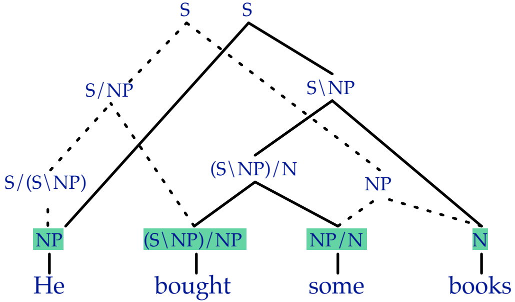
But this doesn't carry over to an oracle forest



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

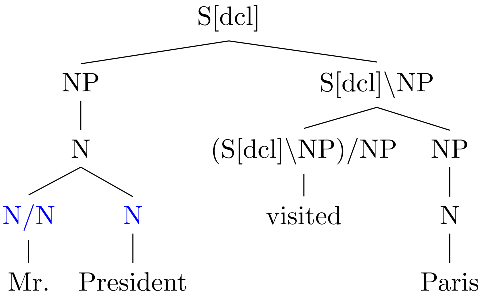
The Dependency Model Oracle

But this doesn't carry over to an oracle forest



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

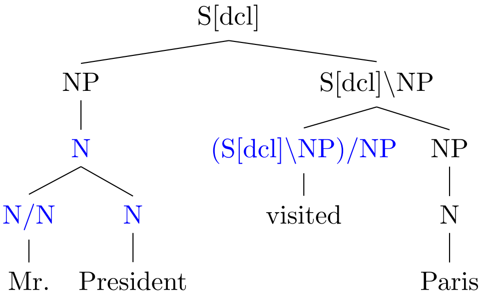
The Dependency Model Oracle

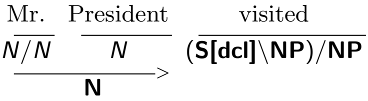


Mr. President

N/N **N**

The Dependency Model Oracle





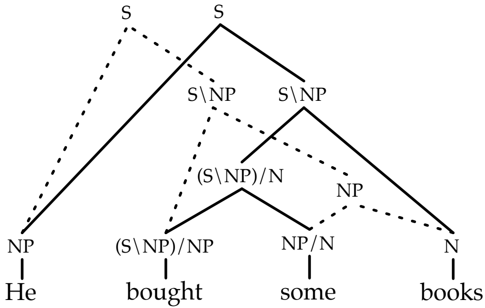
The Dependency Model Oracle

- The dependency oracle

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

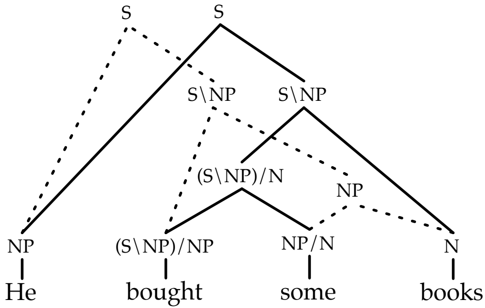
- Shared ancestor set
  - contains possible valid nodes an item should visit
  - is built on-the-fly during decoding for each action type
  - constructed with each *valid* action

The Dependency Model



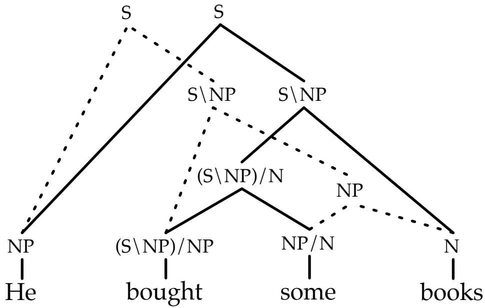
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$

The Dependency Model



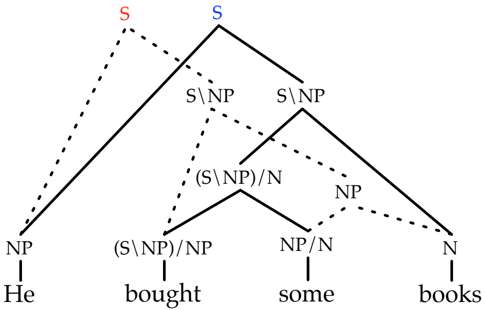
stack ($s_n, ..., s_1, s_0$)	$\mathcal{R}(c_{s_0})$
NP	

The Dependency Model



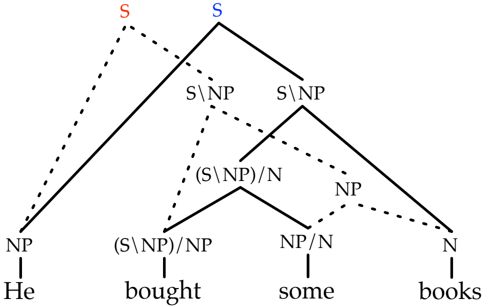
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	()

The Dependency Model



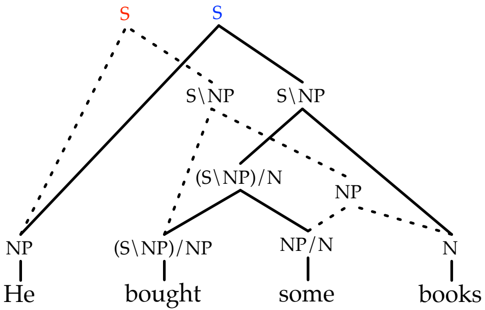
stack ($s_n, ..., s_1, s_0$)	$\mathcal{R}(c_{s_0})$
NP	()
$NP\ (S\backslash NP)/NP$	

The Dependency Model



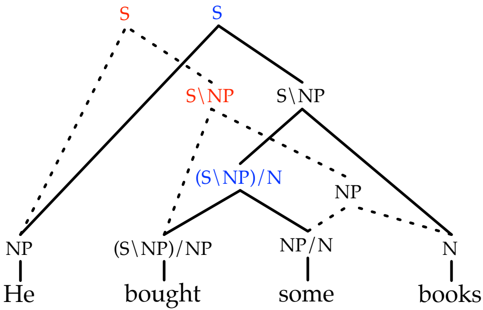
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)

The Dependency Model



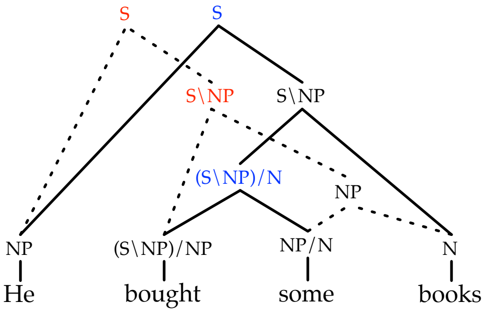
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	()
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	

The Dependency Model



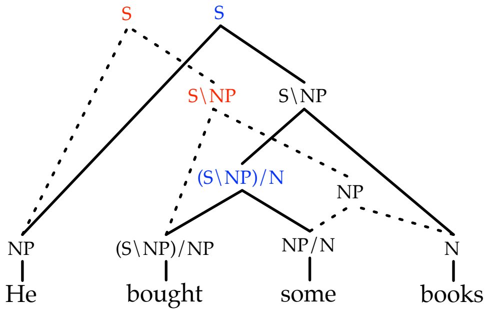
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	

The Dependency Model



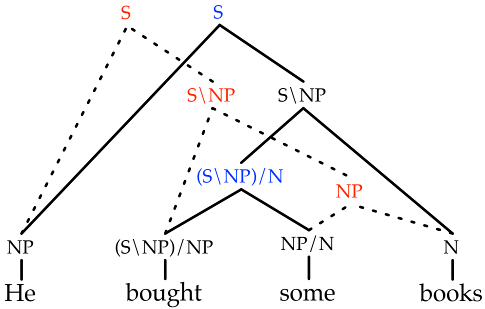
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$

The Dependency Model



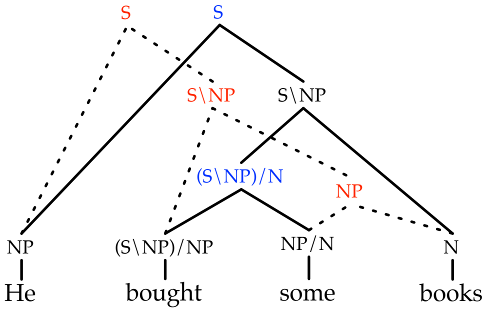
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	

The Dependency Model



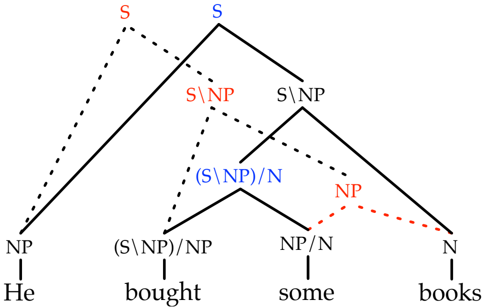
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	

The Dependency Model



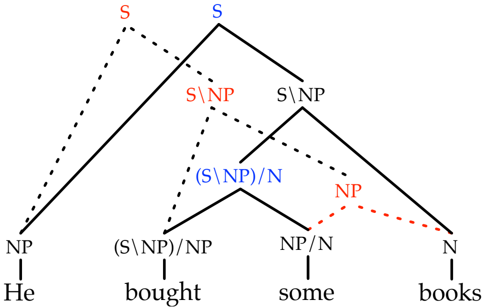
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)

The Dependency Model



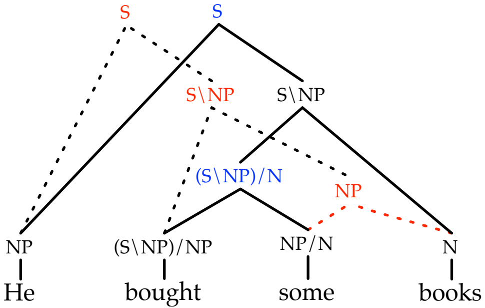
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	

The Dependency Model



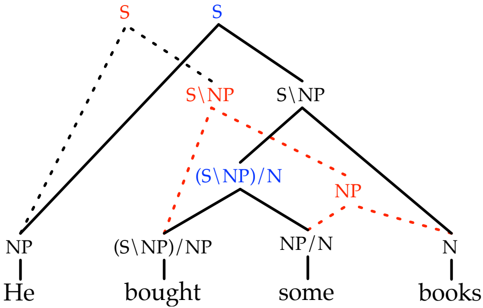
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$ ◄
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	

The Dependency Model



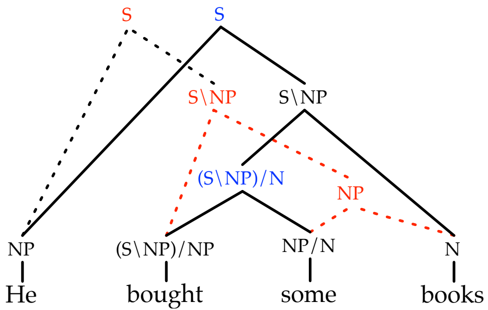
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$

The Dependency Model



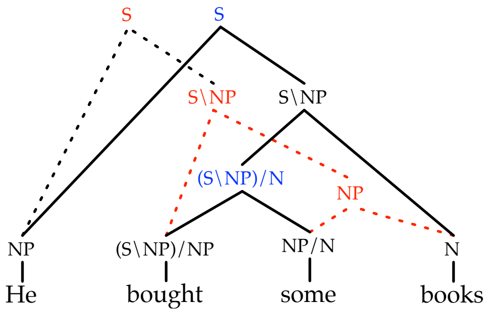
stack $(s_n, \ldots, s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S)
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$
$NP\ S\backslash NP$	

The Dependency Model



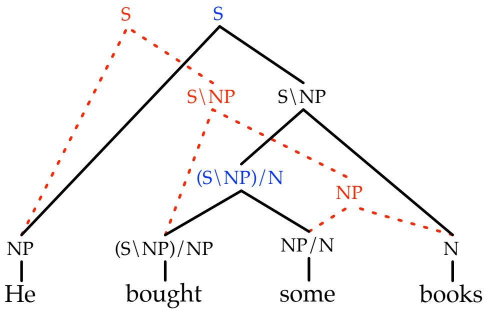
stack $(s_n, ..., s_1, s_0)$	$\mathcal{R}(c_{s_0})$
NP	$()$
$NP\ (S\backslash NP)/NP$	(S, S) ◀
$NP\ (S\backslash NP)/NP\ NP/N$	$(S\backslash NP, (S\backslash NP)/N)$
$NP\ (S\backslash NP)/NP\ NP/N\ N$	(NP)
$NP\ (S\backslash NP)/NP\ NP$	$(S\backslash NP)$
$NP\ S\backslash NP$	

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

Training: Chart-based Dependency Model

- Exponentially many derivations ω consistent with a dependency structure π [Clark and Curran, 2007]

$$P(\pi|S) = \sum_{\omega \in \Delta(\pi)} P(\omega, \pi|S)$$

Training: Chart-based Dependency Model

- Exponentially many derivations ω consistent with a dependency structure π [Clark and Curran, 2007]

$$P(\pi|S) = \sum_{\omega \in \Delta(\pi)} P(\omega, \pi|S)$$

$$\begin{aligned}
L'(\Lambda) &= L(\Lambda) - G(\Lambda) \\
&= \log \prod_{j=1}^{m} P_\Lambda(\pi_j|S_j) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\
&= \sum_{j=1}^{m} \log \frac{\sum_{d \in \Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)}}{\sum_{\omega \in \rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)}} - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2}
\end{aligned}$$

Training: Chart-based Dependency Model

- Exponentially many derivations ω consistent with a dependency structure π [Clark and Curran, 2007]

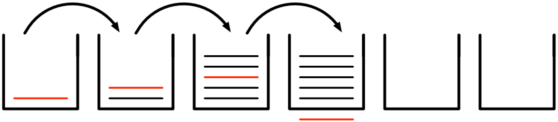
$$P(\pi|S) = \sum_{\omega \in \Delta(\pi)} P(\omega, \pi|S)$$

$$\begin{aligned}
L'(\Lambda) &= L(\Lambda) - G(\Lambda) \\
&= \log \prod_{j=1}^{m} P_\Lambda(\pi_j|S_j) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \\
&= \sum_{j=1}^{m} \log \frac{\sum_{d \in \Delta(\pi_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(d,\pi_j)}}{\sum_{\omega \in \rho(S_j)} e^{\boldsymbol{\lambda}.\boldsymbol{f}(\omega)}} - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2}
\end{aligned}$$

- Requires summing over all ω

Online Training

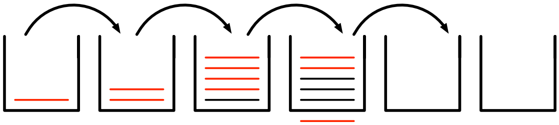
- The normal-form model uses the perceptron with early update
  - only one correct sequence
  - "violation" is guaranteed [Huang et al., 2012]



$$y^* \leftarrow \underset{y \in \mathrm{GEN}(x_i)}{\arg\max}\ \mathbf{w} \cdot \Phi(x_i, y)$$

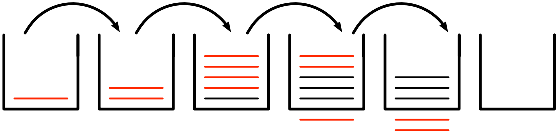
Online Training

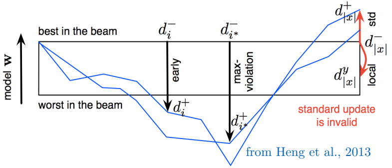
- Standard early update no longer valid for the dependency model
  - multiple correct items possible in each beam
  - "violation" is *not* guaranteed [Huang et al, 2012]



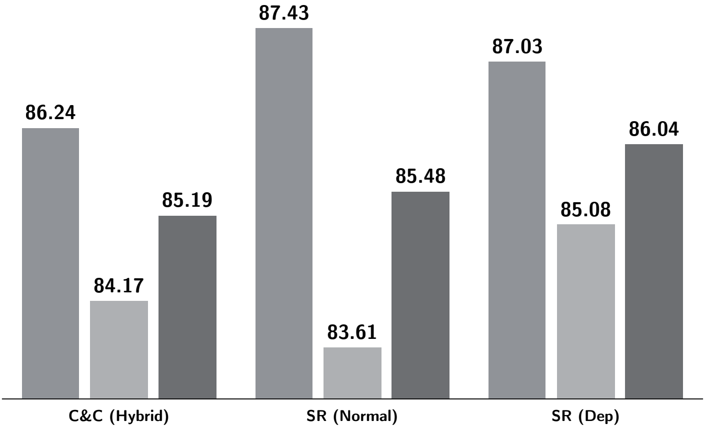
Online Training

- 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]
  - $\mathbf{w} \leftarrow \mathbf{w} + \phi(\Pi_G[0]) - \phi(\mathcal{B}_i[0])$

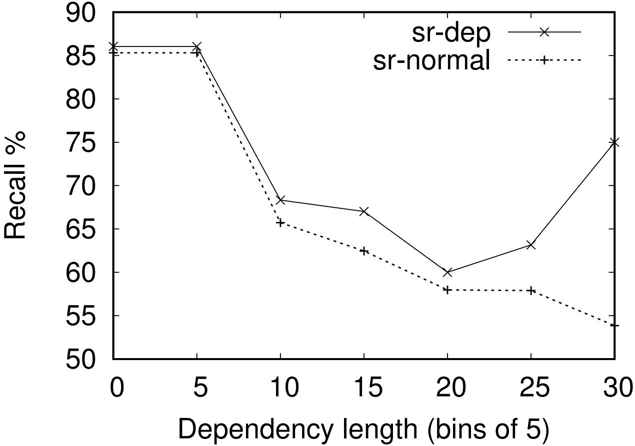




Results


86.24
84.17
85.19
87.43
83.61
85.48
87.03
85.08
86.04
C&C (Hybrid)
SR (Normal)
SR (Dep)

Results



sr-dep
sr-normal
Recall %
90
85
80
75
70
65
60
55
50
0
5
10
15
20
25
30
Dependency length (bins of 5)

Model 2

[Xu et al., NAACL 2016]

Shift-Reduce Parsing

- Linear model (struct. perceptron, SVM etc.)
  - $score(y_i) = \mathbf{w} \cdot \phi(\langle s, q \rangle, y_i)$
  - $score(y) = \sum_{i=1}^{|y|} score(y_i)$
  - $y^* = \arg\max_{y \in \mathcal{Y}_x} score(y)$

Shift-Reduce Parsing

- Linear model (struct. perceptron, SVM etc.)
  - $score(y_i) = \mathbf{w} \cdot \phi(\langle s, q \rangle, y_i)$
  - $score(y) = \sum_{i=1}^{|y|} score(y_i)$
  - $y^* = \arg\max_{y \in \mathcal{Y}_x} score(y)$
- Great flexibility in defining the feature functions

Shift-Reduce Parsing

- Linear model (struct. perceptron, SVM etc.)
  - $score(y_i) = \mathbf{w} \cdot \phi(\langle s, q \rangle, y_i)$
  - $score(y) = \sum_{i=1}^{|y|} score(y_i)$
  - $y^* = \arg\max_{y \in \mathcal{Y}_x} score(y)$
- Great flexibility in defining the feature functions
  - results in millions of indicator features

Shift-Reduce Parsing

- Linear model (struct. perceptron, SVM etc.)
  - $score(y_i) = \mathbf{w} \cdot \phi(\langle s, q \rangle, y_i)$
  - $score(y) = \sum_{i=1}^{|y|} score(y_i)$
  - $y^* = \underset{y \in \mathcal{Y}_x}{\arg\max}\ score(y)$

- Great flexibility in defining the feature functions
  - results in millions of indicator features
  - sparse and expensive to compute

Shift-Reduce Parsing

- Linear model (struct. perceptron, SVM etc.)
  - $score(y_i) = \mathbf{w} \cdot \phi(\langle s, q \rangle, y_i)$
  - $score(y) = \sum_{i=1}^{|y|} score(y_i)$
  - $y^* = \arg\max_{y \in \mathcal{Y}_x} score(y)$
- Great flexibility in defining the feature functions
  - results in millions of indicator features
  - sparse and expensive to compute
- [Yamada and Matsumoto, 2003; Huang and Sagae, 2010; Zhang and Clark, 2011; Zhang and Nivre, 2011; Goldberg and Nivre, 2012; Bohnet et al., 2013; Zhu et al., 2013]

Sparse Features

	feature templates
1	S_0wp, S_0c, S_0pc, S_0wc, S_1wp, S_1c, S_1pc, S_1wc, S_2pc, S_2wc, S_3pc, S_3wc,
2	Q_0wp, Q_1wp, Q_2wp, Q_3wp,
3	S_0Lpc, S_0Lwc, S_0Rpc, S_0Rwc, S_0Upc, S_0Uwc, S_1Lpc, S_1Lwc, S_1Rpc, S_1Rwc, S_1Upc, S_1Uwc,
4	S_0wcS_1wc, S_0cS_1w, S_0wS_1c, S_0cS_1c, S_0wcQ_0wp, S_0cQ_0wp, S_0wcQ_0p, S_0cQ_0p, S_1wcQ_0wp, S_1cQ_0wp, S_1wcQ_0p, S_1cQ_0p,
5	$S_0wcS_1cQ_0p$, $S_0cS_1wcQ_0p$, $S_0cS_1cQ_0wp$, $S_0cS_1cQ_0p$, $S_0pS_1pQ_0p$, $S_0wcQ_0pQ_1p$, $S_0cQ_0wpQ_1p$, $S_0cQ_0pQ_1wp$, $S_0cQ_0pQ_1p$, $S_0pQ_0pQ_1p$, $S_0wcS_1cS_2c$, $S_0cS_1wcS_2c$, $S_0cS_1cS_2wc$, $S_0cS_1cS_2c$, $S_0pS_1pS_2p$,
6	$S_0cS_0HcS_0Lc$, $S_0cS_0HcS_0Rc$, $S_1cS_1HcS_1Rc$, $S_0cS_0RcQ_0p$, $S_0cS_0RcQ_0w$, $S_0cS_0LcS_1c$, $S_0cS_0LcS_1w$, $S_0cS_1cS_1Rc$, $S_0wS_1cS_1Rc$.

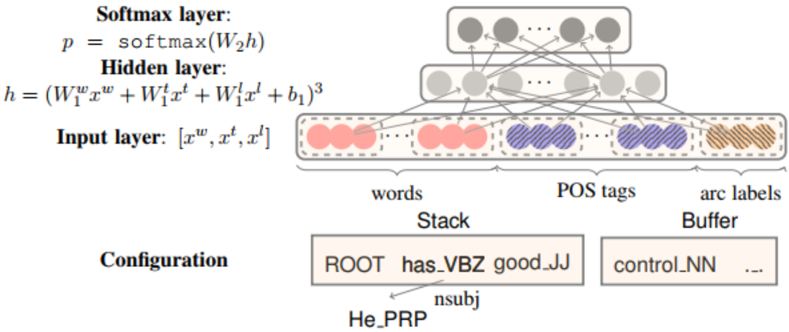
Table 1: Feature templates.

[Zhang and Clark, 2011]

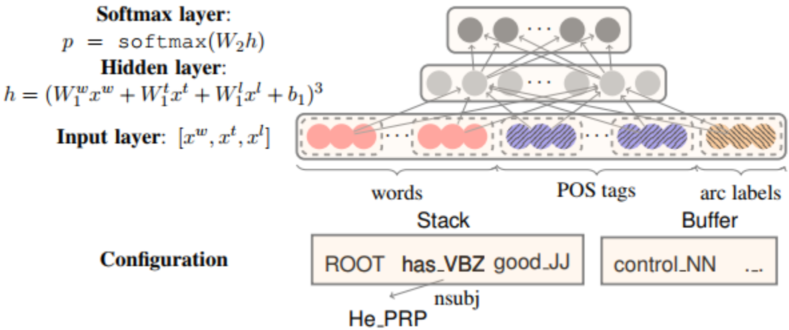
Kernel Features [Chen and Manning, 2014]

s_0.w	s_1.w	s_2.w	s_3.w
s.w$_0$	$s_.$w$_1$	s.w$_2$	s.w$_3$
s_0.l.w	s_1.l.w	s_o.r.w	s_1.r.w
q_0.w	q_1.w	q_2.w	q_3.w
s_0.c	s_0.l.c	s_0.r.c	
s_1.c	s_1.l.c	s_1.r.c	
s_2.c	s_3.c		

Kernel Features [Chen and Manning, 2014]



Kernel Features [Chen and Manning, 2014]



State-of-the-art results at the time!

Local Normalization

$$p(y_t | \langle s, q \rangle_y^{t-1}; \theta) = \frac{\exp\{\gamma(y_t, \langle s, q \rangle_y^{t-1}; \theta)\}}{Z_L\left(\langle s, q \rangle_y^{t-1}\right)}$$

Local Normalization

$$p(y_t|\langle s, q\rangle_y^{t-1}; \theta) = \frac{\exp\{\gamma(y_t, \langle s, q\rangle_y^{t-1}; \theta)\}}{Z_L\left(\langle s, q\rangle_y^{t-1}\right)}$$

$$Z_L(\langle \alpha, \beta\rangle_y^{t-1}) = \sum_{{y_t}' \in \mathcal{T}\left(\langle \alpha, \beta\rangle_y^{t-1}\right)} \exp\{\gamma({y_t}', \langle \alpha, \beta\rangle_y^{t-1}; \theta)\}$$

Local Normalization

$$p(y_t|\langle s,q\rangle_y^{t-1};\theta) = \frac{\exp\{\gamma(y_t,\langle s,q\rangle_y^{t-1};\theta)\}}{Z_L\left(\langle s,q\rangle_y^{t-1}\right)}$$

$$Z_L(\langle \alpha,\beta\rangle_y^{t-1}) = \sum_{{y_t}'\in\mathcal{T}\left(\langle\alpha,\beta\rangle_y^{t-1}\right)} \exp\{\gamma({y_t}',\langle\alpha,\beta\rangle_y^{t-1};\theta)\}$$

$$p(y|\theta) = \prod_{t=1}^{|y|} p(y_t|(\langle\alpha,\beta\rangle_y^{t-1});\theta) = \frac{\exp\{\sum_{t=1}^{|y|}\gamma(y_t,\langle\alpha,\beta\rangle_y^{t-1};\theta)\}}{\prod_{t=1}^{|y|} Z_L(\langle\alpha,\beta\rangle_y^{t-1})}$$

Global Normalization (CRF)

$$p(y|\theta) = \frac{\exp\{\sum_{t=1}^{|y|} \gamma(y_t, \langle \alpha, \beta \rangle_y^{t-1}; \theta)\}}{Z_G}$$

Global Normalization (CRF)

$$p(y|\theta) = \frac{\exp\{\sum_{t=1}^{|y|} \gamma(y_t, \langle \alpha, \beta \rangle_y^{t-1}; \theta)\}}{Z_G}$$

$$Z_G = \sum_{y' \in \mathcal{S}_{|y|}} \exp \sum_{t=1}^{|y|} \gamma(y'_t, \langle \alpha, \beta \rangle_{y'}^{t-1}; \theta)$$

Global Normalization (CRF)

$$p(y|\theta) = \frac{\exp\{\sum_{t=1}^{|y|} \gamma(y_t, \langle \alpha, \beta \rangle_y^{t-1}; \theta)\}}{Z_G}$$

$$Z_G = \sum_{y' \in \mathcal{S}_{|y|}} \exp \sum_{t=1}^{|y|} \gamma(y'_t, \langle \alpha, \beta \rangle_{y'}^{t-1}; \theta)$$

$$\boxed{y^* = \arg\max_{y' \in \mathcal{S}_{|y|}} \sum_{t=1}^{|y|} \gamma(y'_t, \langle \alpha, \beta \rangle_{y'}^{t-1}; \theta)}$$

[Zhou et al., 2015; Andor et al., 2016]

Local vs. Global Normalization

$$Z_L(\langle\alpha,\beta\rangle_y^{t-1}) = \sum_{y_t{}' \in \mathcal{T}\left(\langle\alpha,\beta\rangle_y^{t-1}\right)} \exp\{\gamma(y_t{}', \langle\alpha,\beta\rangle_y^{t-1}; \theta)\}$$

$$Z_G = \sum_{y' \in \mathcal{S}_{|y|}} \exp \sum_{t=1}^{|y|} \gamma(y'_t, \langle\alpha,\beta\rangle_{y'}^{t-1}; \theta)$$

The label bias problem [Bottou et al., 1997; Le Cun et al., 1998; Lafferty et al., 2001];
Andor et al., 2016 showed that $\mathcal{P}_L \subset \mathcal{P}_G$ (assuming no lookahead)

Expected F-measure Training for Shift-Reduce Parsing with RNNs

	NN	Beam (Train)	Beam (Test)	global
C&M, 2014	✓	✗	✓	✗

Expected F-measure Training for Shift-Reduce Parsing with RNNs

	NN	Beam (Train)	Beam (Test)	global
C&M, 2014	✓	✗	✓	✗
present model	✓	✓	✓	✓

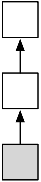
Expected F-measure Training for Shift-Reduce Parsing with RNNs

	NN	Beam (Train)	Beam (Test)	global
C&M, 2014	✓	✗	✓	✗
present model	✓	✓	✓	✓

At the same time, the model is optimized towards the final evaluation metric ✓

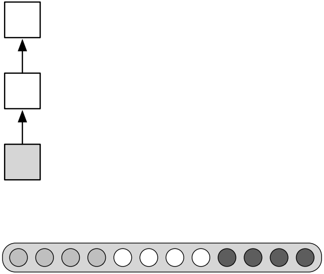
Expected F-Measure Training: Step 1

❶ Train a baseline model using a cross-entropy loss (pretraining)



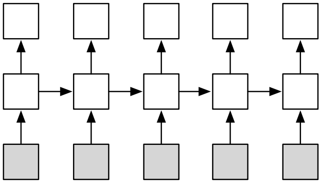
Expected F-Measure Training: Step 1

1. Train a baseline model using a cross-entropy loss (pretraining)



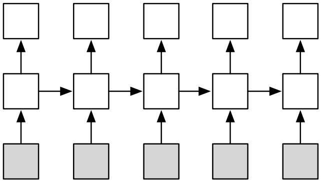
Expected F-Measure Training: Step 1

1. Train a baseline model using a cross-entropy loss (pretraining)



Expected F-Measure Training: Step 1

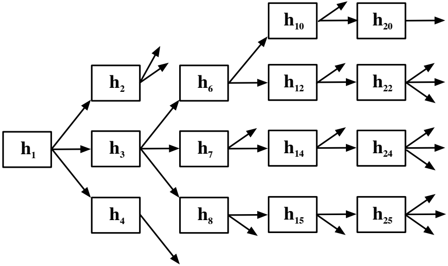
❶ Train a baseline model using a cross-entropy loss (pretraining)



$$L(\theta_B) = -\sum_{k}^{T_i} p(t_k|\theta_B), \qquad \theta_B = \{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$$

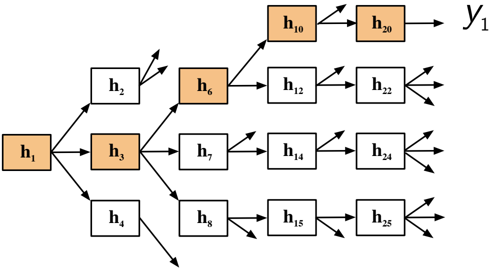
Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



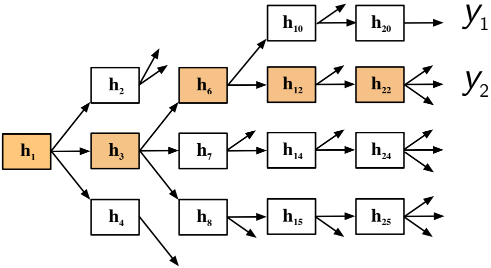
Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



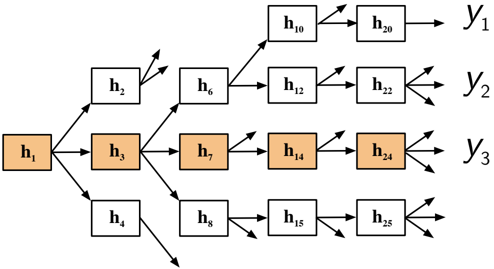
Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



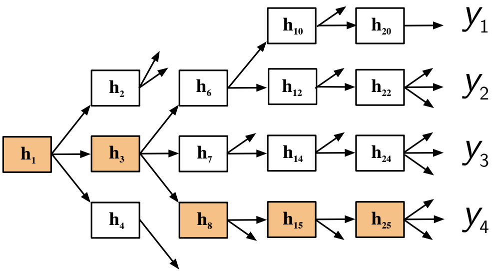
Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



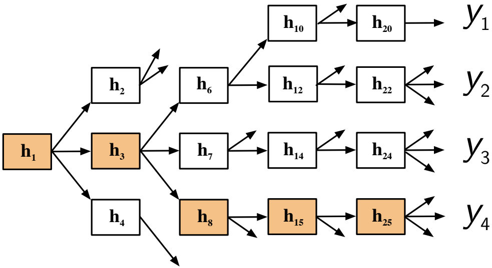
Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



Expected F-Measure Training: Step 2

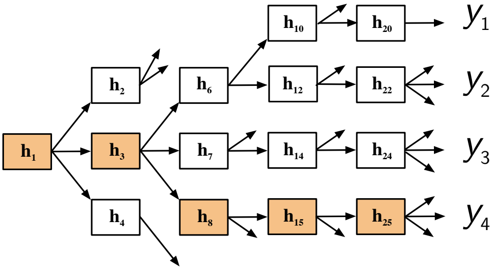
❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



$$\gamma(y_i) = \sum_{j=1}^{|y_i|} \log s_\theta(y_{ij})$$

Expected F-Measure Training: Step 2

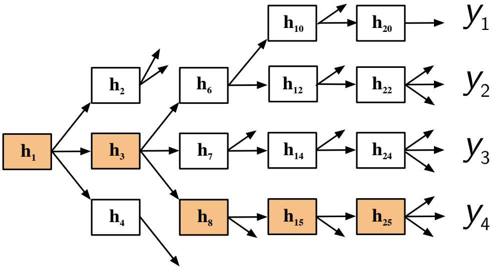
❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



$$\gamma(y_i) = \sum_{j=1}^{|y_i|} \log s_\theta(y_{ij}), \qquad \text{F1}(\Delta_{y_i}, \Delta_{x_n}^G)$$

Expected F-Measure Training: Step 2

❷ Let $\theta = \theta_B$, and parse each sentence in the training data with beam search



$$\gamma(y_i) = \sum_{j=1}^{|y_i|} \log s_\theta(y_{ij}), \qquad \mathrm{F1}(\Delta_{y_i}, \Delta_{x_n}^{G})$$

Expected F-Measure Training: Step 3

3. Compute the -XF1 loss for each sentence, do SGD update and iterate

$$J(\theta) = -\text{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\text{F1}(\Delta_{y_i}, \Delta_{x_n}^G),$$

$$p(y_i|\theta) = \frac{\exp\{\gamma(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\gamma(y)\}}$$

Expected F-Measure Training: Step 3

❸ Compute the -XF1 loss for each sentence, do SGD update and iterate

$$J(\theta) = -\mathrm{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\mathrm{F1}(\Delta_{y_i}, \Delta_{x_n}^{G}),$$

$$p(y_i|\theta) = \frac{\exp\{\gamma(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\gamma(y)\}}$$

Expected F-Measure Training: Step 3

❸ Compute the -XF1 loss for each sentence, do SGD update and iterate

$$J(\theta) = -\text{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\text{F1}(\Delta_{y_i}, \Delta^G_{x_n}),$$

$$p(y_i|\theta) = \frac{\exp\{\gamma(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\gamma(y)\}}$$

$$\frac{\partial J(\theta)}{\partial \theta} = -\sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \frac{\partial J(\theta)}{\partial s_\theta(y_{ij})} \frac{\partial s_\theta(y_{ij})}{\partial \theta}$$

Expected F-Measure Training: Step 3

❸ Compute the -XF1 loss for each sentence, do SGD update and iterate

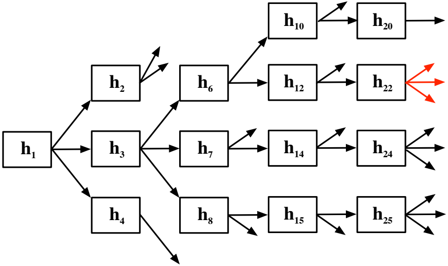
$$J(\theta) = -\text{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\text{F1}(\Delta_{y_i}, \Delta_{x_n}^G),$$

$$p(y_i|\theta) = \frac{\exp\{\gamma(y_i)\}}{\sum_{y \in \Lambda(x_n)} \exp\{\gamma(y)\}}$$

$$\frac{\partial J(\theta)}{\partial \theta} = -\sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \frac{\partial J(\theta)}{\partial s_\theta(y_{ij})} \frac{\partial s_\theta(y_{ij})}{\partial \theta}$$

Expected F-Measure Training: Step 3

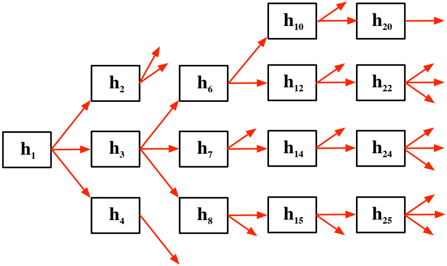
❸ Compute the -XF1 loss for each sentence, do SGD update and iterate



$$\frac{\partial J(\theta)}{\partial \theta} = - \sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \frac{\partial J(\theta)}{\partial s_\theta(y_{ij})} \frac{\partial s_\theta(y_{ij})}{\partial \theta}$$

Expected F-Measure Training: Step 3

❸ Compute the -XF1 loss for each sentence, do SGD update and iterate



$$\frac{\partial J(\theta)}{\partial \theta} = -\sum_{y_i \in \Lambda(x_n)} \sum_{y_{ij} \in y_i} \frac{\partial J(\theta)}{\partial s_\theta(y_{ij})} \frac{\partial s_\theta(y_{ij})}{\partial \theta}$$

Expected F-Measure Training

output	action sequence	$\gamma(y_i)$	F1
y_1	$y_{11}\ y_{12} \dots y_{1i}$	-0.60	0.67
y_2	$y_{21}\ y_{22} \dots y_{2j}$	-1.5	0.81
y_3	$y_{31}\ y_{32} \dots y_{3k}$	-4.96	0.90

Expected F-Measure Training

output	action sequence	$\gamma(y_i)$	F1
y_1	$y_{11}\ y_{12} \dots y_{1i}$	-0.60	0.67
y_2	$y_{21}\ y_{22} \dots y_{2j}$	-1.5	0.81
y_3	$y_{31}\ y_{32} \dots y_{3k}$	-4.96	0.90

$$J(\theta) = -\mathrm{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\mathrm{F1}(\Delta_{y_i}, \Delta_{x_n}^{G}) = -71.00$$

Expected F-Measure Training

output	action sequence	$\gamma(y_i)$	F1
y_1	$y_{11}\ y_{12} \ldots y_{1i}$	-0.60	0.67
y_2	$y_{21}\ y_{22} \ldots y_{2j}$	-1.5	0.81
y_3	$y_{31}\ y_{32} \ldots y_{3k}$	-4.96	0.90

$$J(\theta) = -\mathrm{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\mathrm{F1}(\Delta_{y_i}, \Delta^G_{x_n}) = -71.00$$

output	action sequence	$\gamma(y_i)$	F1
z_1	$z_{11}\ z_{12} \ldots z_{1i}$	-0.90	0.71
z_2	$z_{21}\ z_{22} \ldots z_{2j}$	-0.99	0.72
z_3	$z_{31}\ z_{32} \ldots z_{3k}$	-3.76	0.73

Expected F-Measure Training

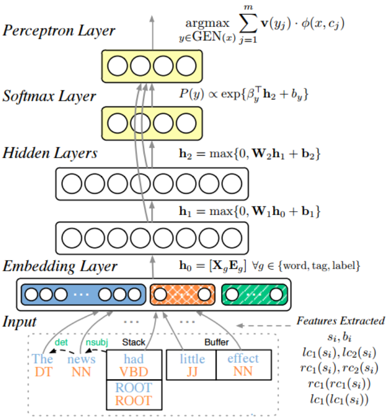
output	action sequence	$\gamma(y_i)$	F1
y_1	$y_{11}\ y_{12} \ldots y_{1i}$	-0.60	0.67
y_2	$y_{21}\ y_{22} \ldots y_{2j}$	-1.5	0.81
y_3	$y_{31}\ y_{32} \ldots y_{3k}$	-4.96	0.90

$$J(\theta) = -\mathrm{XF1}(\theta) = -\sum_{y_i \in \Lambda(x_n)} p(y_i|\theta)\mathrm{F1}(\Delta_{y_i}, \Delta^G_{x_n}) = -71.00$$

output	action sequence	$\gamma(y_i)$	F1
z_1	$z_{11}\ z_{12} \ldots z_{1i}$	-0.90	0.71
z_2	$z_{21}\ z_{22} \ldots z_{2j}$	-0.99	0.72
z_3	$z_{31}\ z_{32} \ldots z_{3k}$	-3.76	0.73

$$J(\theta) = -\mathrm{XF1}(\theta) = -\sum_{z_i \in \Lambda(x_n)} p(z_i|\theta)\mathrm{F1}(\Delta_{z_i}, \Delta^G_{x_n}) = -71.20$$

Related Work



[Weiss et al., 2015]

Related Work

- Watanabe and Sumita, 2015
  - max-margin based objective
  - max-violation updates [Huang et al., 2012]

- Zhou et al., 2015
  - based on Chen and Manning, 2014
  - CRF [Bottou et al., 1997; Le Cun et al., 1998; Lafferty et al., 2001]

- Andor et al., 2016
  - based on Chen and Manning, 2014 and Weiss et al., 2015
  - also CRF

Related Work

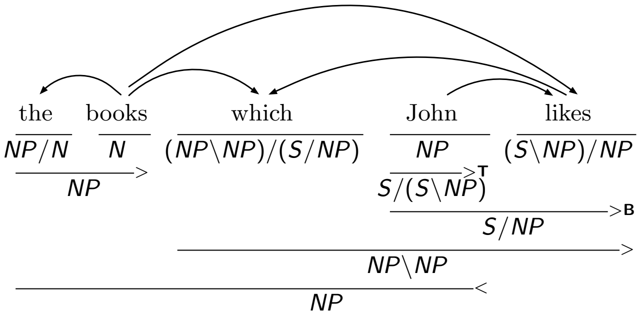
- Watanabe and Sumita, 2015
  - max-margin based objective
  - max-violation updates [Huang et al., 2012]

- Zhou et al., 2015
  - based on Chen and Manning, 2014
  - CRF [Bottou et al., 1997; Le Cun et al., 1998; Lafferty et al., 2001]

- Andor et al., 2016
  - based on Chen and Manning, 2014 and Weiss et al., 2015
  - also CRF

- Optimizing task-specific metrics for parsing
  - e.g., Goodman, 1996; Smith and Eisner, 2006; Auli and Lopez, 2011

Eval: F1 over Labeled, Directed CCG Deps



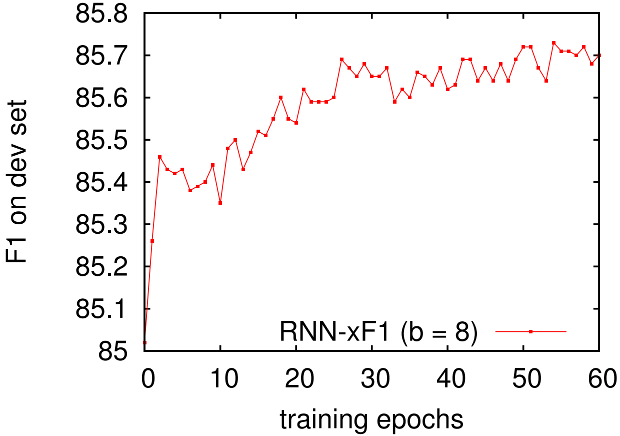
$$\langle \textit{the},\ NP/N_1,\ 1,\ \textit{books},\rangle$$
$$\langle \textit{likes},\ (S\backslash NP_1)/NP_2,\ 1,\ \textit{John}\rangle$$
$$\langle \textit{which},\ (NP/NP_1)/(S/NP)_2,\ 2,\ \textit{likes}\rangle$$
$$\langle \textit{which},\ (NP/NP_1)/(S/NP)_2,\ 1,\ \textit{books}\rangle$$
$$\langle \textit{likes},\ (S\backslash NP_1)/NP_2,\ 2,\ \textit{books}\rangle$$

The Greedy Model and Beam Search (Dev)

beam	F1
$b = 1$	84.61
$b = 2$	84.94
$b = 4$	85.01
$b = 6$	85.02
$b = 8$	85.02
$b = 16$	85.01

$b \in \{6, 8\}$ gives $+0.41\%$ F1 over $b = 1$

XF1 Model Dev F1 vs. Training Epochs



Test Set Parsing Results

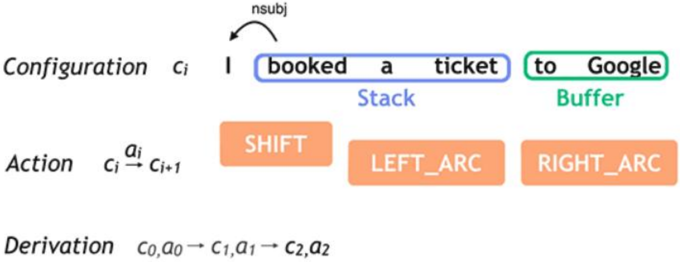
Model	LP	LR	LF	CAT	Speed
C&C (normal)	85.45	83.97	84.70	92.83	97.90
C&C (hybrid)	86.24	84.17	85.19	93.00	95.25
Zhang11 ($b = 16$)	87.04	84.14	85.56	92.95	49.54
Xu14 ($b = 128$)	87.03	85.08	86.04	93.10	12.85
Am16 ($b = 1$)	-	-	83.27	91.89	350.00
Am16 ($b = 16$)	-	-	85.57	92.86	10.00
RNN-greedy ($b = 1$)	88.53	81.65	84.95	93.57	337.45
RNN-greedy ($b = 6$)	88.54	82.77	85.56	93.68	96.04
RNN-XF1 ($b = 8$)	88.74	84.22	86.42	93.87	67.65

- Zhang11 = Zhang and Clark, 2011*, Xu14 = [Xu et al., 2014]; AM16 = Ambati et al., 2016 (NN + Struct. Percep [Weiss et al., 2015])
- The XF1 model improves LR by 2.57% and LF by 1.47% over RNN-greedy ($b = 1$)

Model 3

[Xu, EMNLP 2016]

Transition-based Dependency Parsing



source: Google SyntaxNet

Models

- Local linear (e.g., SVM)

Models

- Local linear (e.g., SVM) $\Rightarrow$ global linear (e.g., struct. perceptron)

Models

- Local linear (e.g., SVM) $\Rightarrow$ global linear (e.g., struct. perceptron)

- Local NNs and RNNs

Models

- Local linear (e.g., SVM) $\Rightarrow$ global linear (e.g., struct. perceptron)

- Local NNs and RNNs $\Rightarrow$ global NNs and RNNs (e.g., NNs + CRF [Andor et al., 2016] and XF1)

Models

- Local linear (e.g., SVM) $\Rightarrow$ global linear (e.g., struct. perceptron)
- Local NNs and RNNs $\Rightarrow$ global NNs and RNNs (e.g., NNs + CRF [Andor et al., 2016] and XF1)

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

No "global" sensitivity to parser states

Models

- Local linear (e.g., SVM) $\Rightarrow$ global linear (e.g., struct. perceptron)

- Local NNs and RNNs $\Rightarrow$ global NNs and RNNs (e.g., NNs + CRF [Andor et al., 2016] and XF1)

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

No “global” sensitivity to parser states

Solution: Stack-LSTM [Dyer et al., 2015]

Stack-LSTM [Dyer et al., 2015]

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

Stack-LSTM [Dyer et al., 2015]

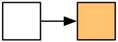
step	stack ($s_n, \ldots, s_1, s_0$)	queue ($q_0, q_1 \ldots, q_n$)	action
0		Ms. Haag plays Elianti	
1	N/N	Haag plays Elianti	SHIFT
2	N/N N	plays Elianti	SHIFT
3	N	plays Elianti	REDUCE
4	NP	plays Elianti	UNARY
5	NP $(S[dcl]\backslash NP)/NP$	Elianti	SHIFT
6	NP $(S[dcl]\backslash NP)/NP$ N		SHIFT
7	NP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

LSTM-stack , LSTM-queue , LSTM-action

Stack-LSTM [Dyer et al., 2015]

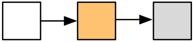
It showed promise

Stack-LSTM [Dyer et al., 2015]



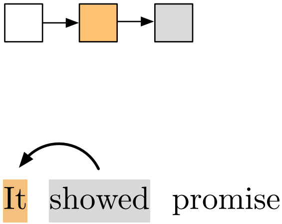
It showed promise

Stack-LSTM [Dyer et al., 2015]

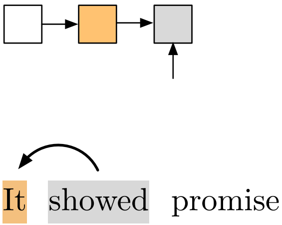


It showed promise

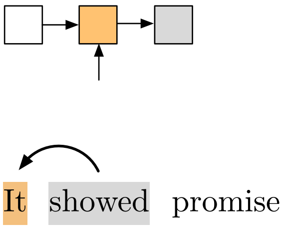
Stack-LSTM [Dyer et al., 2015]



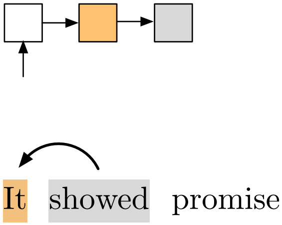
Stack-LSTM [Dyer et al., 2015]



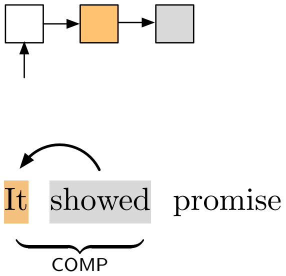
Stack-LSTM [Dyer et al., 2015]



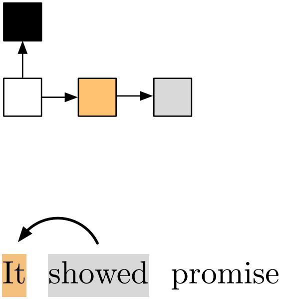
Stack-LSTM [Dyer et al., 2015]



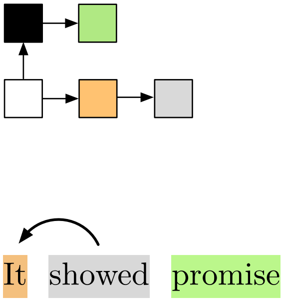
Stack-LSTM [Dyer et al., 2015]



Stack-LSTM [Dyer et al., 2015]

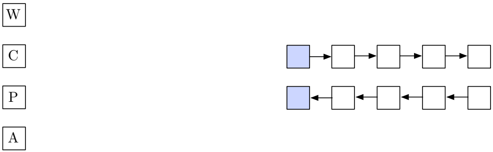


Stack-LSTM [Dyer et al., 2015]



LSTM Shift-Reduce CCG Parsing

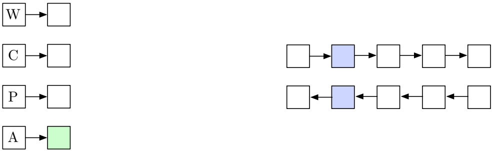
the books which John likes



LSTM Shift-Reduce CCG Parsing

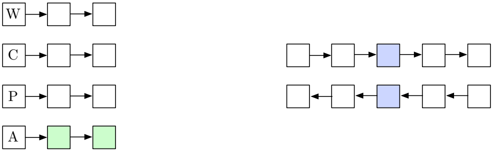
the books which John likes

NP/N



LSTM Shift-Reduce CCG Parsing

the	books	which	John	likes
NP/N	N			



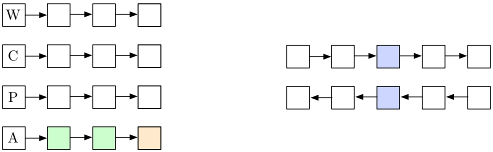
LSTM Shift-Reduce CCG Parsing

the books which John likes

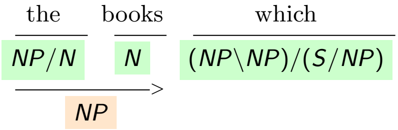
NP/N N

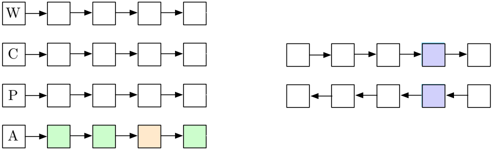
$>$

NP



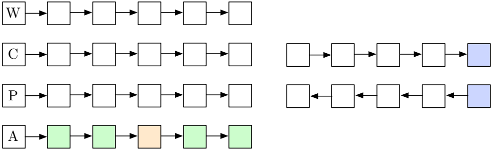
LSTM Shift-Reduce CCG Parsing



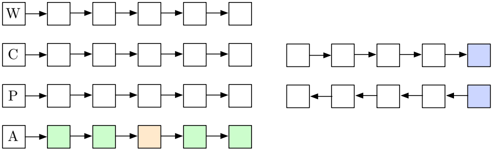


LSTM Shift-Reduce CCG Parsing

the	books	which	John	likes
NP/N	N	$(NP\backslash NP)/(S/NP)$	NP	
NP (>)				

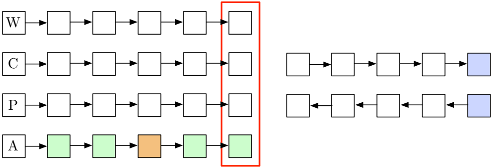


LSTM Shift-Reduce CCG Parsing



LSTM Shift-Reduce CCG Parsing

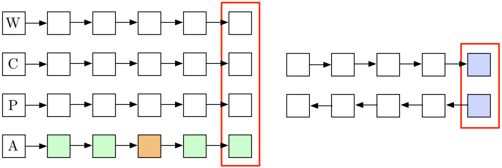
$$\delta_t = [\mathbf{h}_t^{\mathsf{W}}; \mathbf{h}_t^{\mathsf{C}}; \mathbf{h}_t^{\mathsf{P}}; \mathbf{h}_t^{\mathsf{A}}]$$



LSTM Shift-Reduce CCG Parsing

$$\delta_t = [\mathbf{h}_t^{\mathsf{W}}; \mathbf{h}_t^{\mathsf{C}}; \mathbf{h}_t^{\mathsf{P}}; \mathbf{h}_t^{\mathsf{A}}]$$

$$\mathbf{b}_t = f(\mathbf{B}[\delta_t; \mathbf{Q}_j] + \mathbf{r})$$

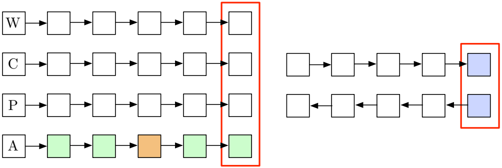


LSTM Shift-Reduce CCG Parsing

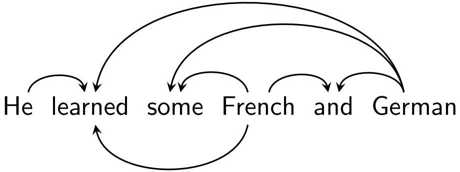
$$\delta_t = [\mathbf{h}_t^{\mathsf{W}}; \mathbf{h}_t^{\mathsf{C}}; \mathbf{h}_t^{\mathsf{P}}; \mathbf{h}_t^{\mathsf{A}}]$$

$$\mathbf{b}_t = f(\mathbf{B}[\delta_t; \mathbf{Q}_j] + \mathbf{r})$$

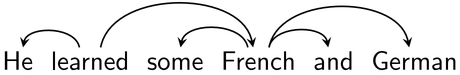
$$\mathbf{a}_t = f(\mathbf{A}\mathbf{b}_t + \mathbf{s})$$



Two Simple Motivations: I



this parser



Google SyntaxNet and Stanford

Two Simple Motivations: II

input : $w_0 \ldots w_{n-1}$

axiom : $0 : (0, \epsilon, \beta, \phi)$

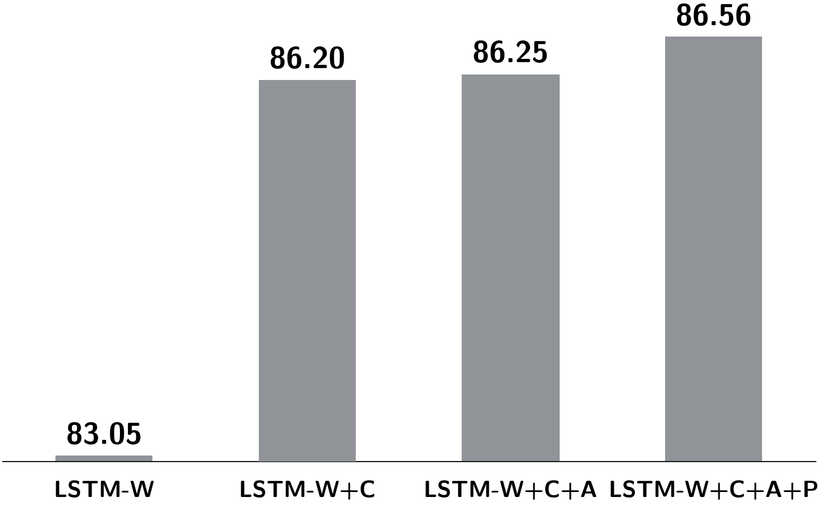
goal : $2n - 1 + \mu : (n, \delta, \epsilon, \Delta)$

$$\frac{\omega : (j, \delta, x_{w_j}|\beta, \Delta)}{\omega + 1 : (j+1, \delta|x_{w_j}, \beta, \Delta)} \qquad (\textsc{shift}; 0 \leq j < n)$$

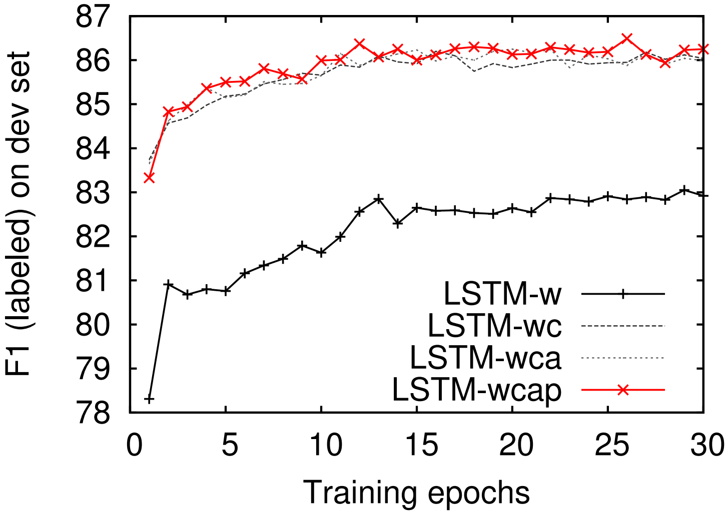
$$\frac{\omega : (j, \delta|s_1|s_0, \beta, \Delta)}{\omega + 1 : (j, \delta|x, \beta, \Delta \cup \langle x \rangle))} \qquad (\textsc{reduce}; s_1 s_0 \rightarrow x)$$

$$\frac{\omega : (j, \delta|s_0, \beta, \Delta)}{\omega + 1 : (j, \delta|x, \beta, \Delta)} \qquad (\textsc{unary}; s_0 \rightarrow x)$$

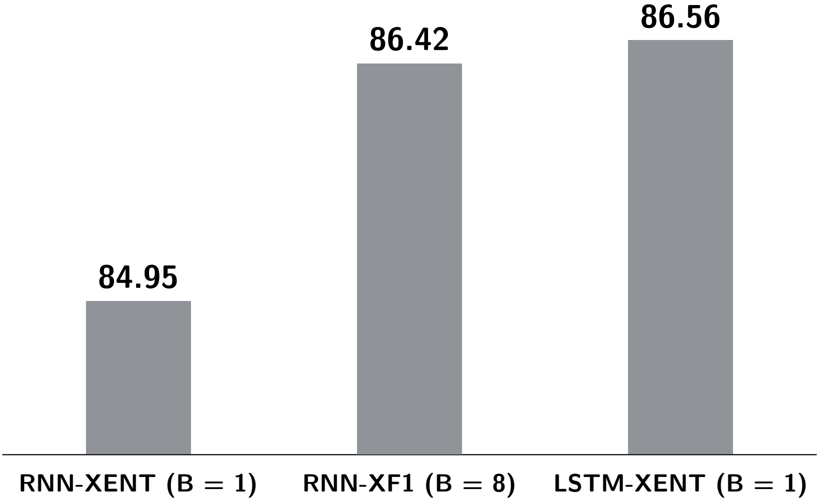
Results: Locally Normalized Models



Results: Locally Normalized Models



Results: Locally Normalized Models



The Label Bias Problem [Bottou et al., 1997; LeCun et al., 1998; Lafferty et al., 2001]

$$p(y_t | \langle s, q \rangle_y^{t-1}; \theta) = \frac{\exp\{\gamma(y_t, \langle s, q \rangle_y^{t-1}; \theta)\}}{Z_L\left(\langle s, q \rangle_y^{t-1}\right)}$$

$$Z_L(\langle \alpha, \beta \rangle_y^{t-1}) = \sum_{y_t{}' \in \mathcal{T}\left(\langle \alpha, \beta \rangle_y^{t-1}\right)} \exp\{\gamma(y_t{}', \langle \alpha, \beta \rangle_y^{t-1}; \theta)\}$$

Andor et al., (2016) showed that $\mathcal{P}_L \subset \mathcal{P}_G$

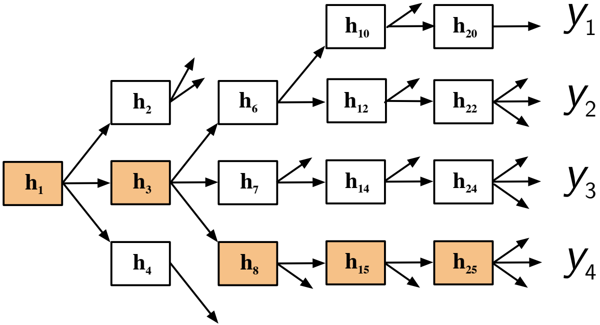
The Label Bias Problem [Bottou et al., 1997; LeCun et al., 1998; Lafferty et al., 2001]

$$p(y_t | \langle s, q \rangle_y^{t-1}; \theta) = \frac{\exp\{\gamma(y_t, \langle s, q \rangle_y^{t-1}; \theta)\}}{Z_L\left(\langle s, q \rangle_y^{t-1}\right)}$$

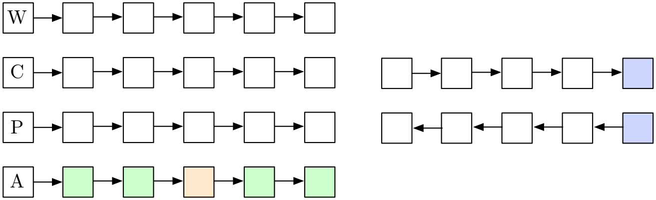
$$Z_L(\langle \alpha, \beta \rangle_y^{t-1}) = \sum_{{y_t}' \in \mathcal{T}\left(\langle \alpha, \beta \rangle_y^{t-1}\right)} \exp\{\gamma({y_t}', \langle \alpha, \beta \rangle_y^{t-1}; \theta)\}$$

Andor et al., (2016) showed that $\mathcal{P}_L \subset \mathcal{P}_G$ and label bias is irrespective of the scoring function γ

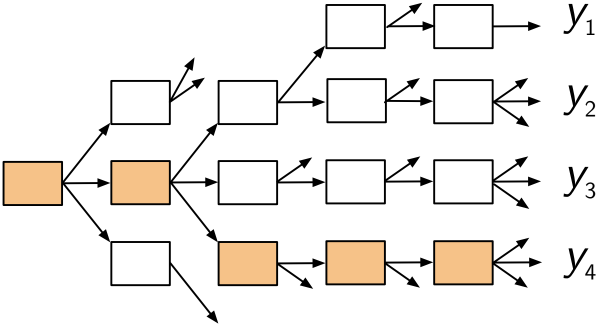
XF1 Training

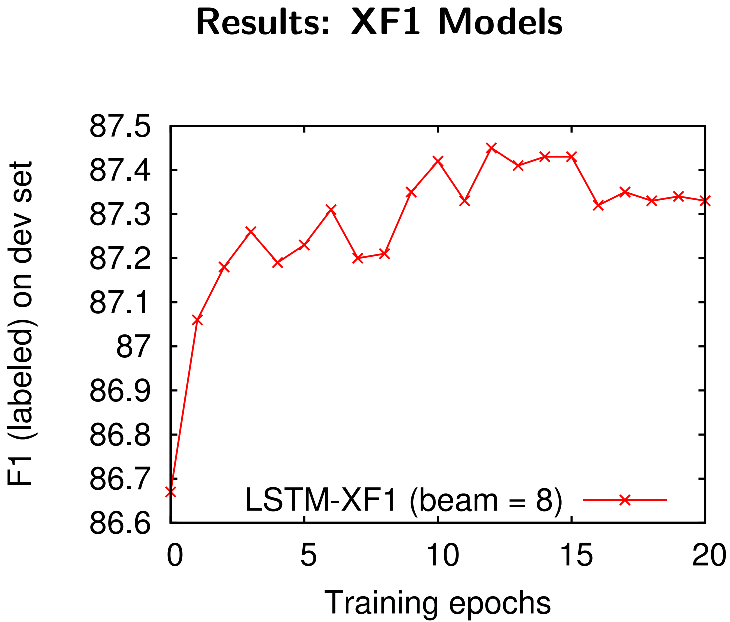


XF1 Training

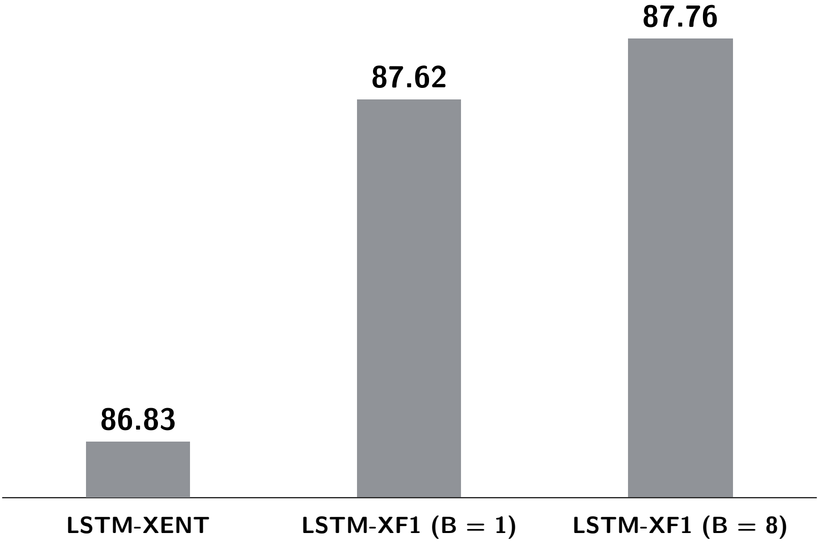


XF1 Training

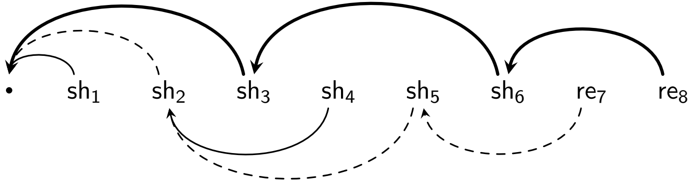



Results: XF1 Models
F1 (labeled) on dev set
87.5
87.4
87.3
87.2
87.1
87
86.9
86.8
86.7
86.6
0
5
10
15
20
Training epochs
LSTM-XF1 (beam = 8)

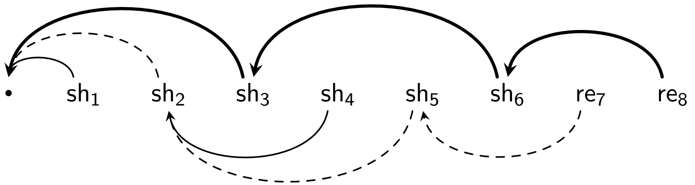
Results: XF1 Models



Impl.: Tree-Structured Stack + Dynamically Structured Graph

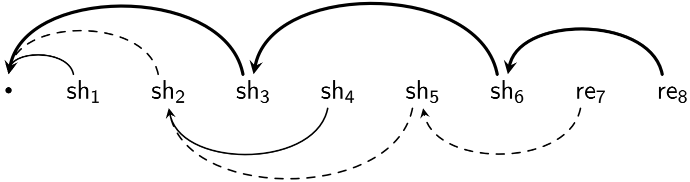


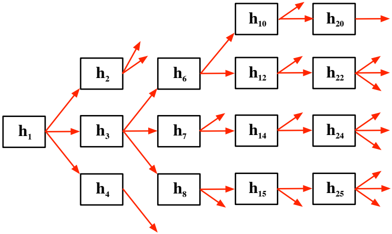
Impl.: Tree-Structured Stack + Dynamically Structured Graph



$$
\begin{aligned}
\boldsymbol{\delta}^{a}_{s_r} &= \mathsf{BPTS}(s_r.\mathcal{A}) \\
&= \sum_{m \in s_r.\mathcal{A}.keys} \sum_{i \in s_r.\mathcal{A}[m]} \boldsymbol{\delta}_m \boldsymbol{\delta}_{im} \\
&= \sum_{m \in s_r.\mathcal{A}.keys} \sum_{i \in s_r.\mathcal{A}[m]} \boldsymbol{\delta}_m \underbrace{p(y_i|\theta)(\mathsf{XF1}(\theta) - \mathsf{F1}(\Delta_{y_i}, \Delta^{G}_{x_n}))\frac{1}{z_m}}_{\text{XF1 gradient per action}}
\end{aligned}
$$

Impl.: Tree-Structured Stack + Dynamically Structured Graph





CNN $\Rightarrow$ DyNet

Conclusions

- Global normal-form

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)

- Local LSTM with global sensitivity

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)

- Local LSTM with global sensitivity $\Rightarrow$ global LSTM (optimized for the evaluation metric)

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)
- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)
- Local LSTM with global sensitivity $\Rightarrow$ global LSTM (optimized for the evaluation metric)
- Beam search

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)

- Local LSTM with global sensitivity $\Rightarrow$ global LSTM (optimized for the evaluation metric)

- Beam search $\Rightarrow$ struct. perceptron, RNN, and LSTM

Conclusions

- Global normal-form $\Rightarrow$ global dependency model with a hidden variable (with the struct. perceptron)

- Local RNN $\Rightarrow$ global RNN (optimized for the evaluation metric)

- Local LSTM with global sensitivity $\Rightarrow$ global LSTM (optimized for the evaluation metric)

- Beam search $\Rightarrow$ struct. perceptron, RNN, and LSTM $\Rightarrow$ global structured learning