Shift-Reduce CCG Parsing

Yue Zhang and Stephen Clark
University of Cambridge

Outline

- Introduction
- CCG
- Our Shift-Reduce Parser
- Experiments

Outline

- Introduction
- CCG
- Our Shift-Reduce Parser
- Experiments

Introduction

- Combinatory Categorial Grammar (CCG)
  - Lexicalized grammar
  - Successfully applied to a range of problems:
    - treebank creation (Hockenmaier and Steedman, 2007)
    - syntactic parsing (Hockenmaier, 2003; Clark and Curran, 2007)
    - logical form construction (Bos et al., 2004)
    - surface realization (White and Rajkumar, 2009)

Introduction

- Combinatory Categorial Grammar (CCG)
  - Binary branching
  - Naturally compatible with bottom-up parsing algorithms, such as shift-reduce and CKY (Ades and Steedman, 1982; Steedman, 2000)
  - Previous work only considered chart-parsing (Clark and Curran, 2007; Hockenmaier, 2003; Fowler and Penn, 2010)
  - We fill a gap in the literature by developing a shift-reduce parser for CCG

Introduction

- Shift-reduce parsing
  - Popular for dependency parsing, following the work of Yamada and Matsumoto (2003) and Nivre and Scholz (2004)
  - Uses a stack and a set of shift-reduce actions to build parses
  - Compared to chart parsing:
    - allows highly efficient parsing
    - allows definition of rich features
    - makes different errors
  - Leads to higher accuracy by combination with chart-parsing

Introduction

- Our shift-reduce parser vs. chart-based C&C
  - C&C (Clark and Curran, 2007) is a chart-based CCG parser
  - Both use global discriminative models
  - Heuristic beam-search with rich features vs. optimal dynamic-programming with restricted features
  - 85.53% F-score vs. 85.45% overall
  - Higher precision and lower recall
  - Naturally handles cases when spanning analysis cannot be found

Outline

- Introduction
- CCG
- Our Shift-Reduce Parser
- Experiments

CCG Parsing

- Lexical categories
  - basic categories: *N* (nouns), *NP* (noun phrases), *PP* (prepositional phrases), ...
  - complex categories: $S\backslash NP$ (intransitive verbs), $(S\backslash NP)/NP$ (transitive verbs), ...
- Adjacent phrases are combined to form larger phrases using category combination e.g.:
  - function application: $NP\ S\backslash NP \Rightarrow S$
  - function composition: $(S\backslash NP)/(S\backslash NP)\ (S\backslash NP)/NP \Rightarrow (S\backslash NP)/NP$
- Unary rules change the type of a phrase
  - Type raising: $NP \Rightarrow S/(S\backslash NP)$
  - Type changing: $S[pss]\backslash NP \Rightarrow NP\backslash NP$

CCG Parsing

- An example derivation

IBM bought Lotus

CCG Parsing

- An example derivation

IBM	bought	Lotus
NP	*(S[dcl]\NP)/NP*	*NP*

CCG Parsing

- An example derivation

IBM	bought	Lotus
NP	*(S[dcl]\NP)/NP*	*NP*

S[dcl]\NP

CCG Parsing

- An example derivation

$$
\dfrac{\text{IBM} \atop NP \qquad \dfrac{\text{bought} \atop (S[dcl]\backslash NP)/NP \qquad \text{Lotus} \atop NP}{S[dcl]\backslash NP}}{S[dcl]}
$$

CCG Parsing

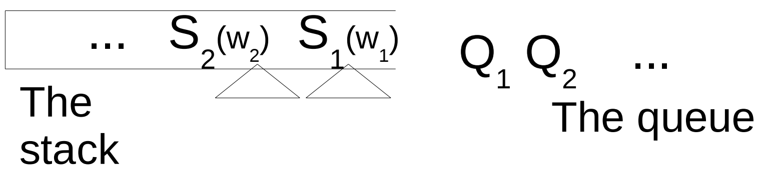
- Rule extraction
  - Manually define the combinatory rule schemas (Steedman, 2000; Clark and Curran, 2007)
    - Schema: $X/Y\ Y \Rightarrow X$
    - Instance: $(S\backslash NP)/NP\ NP \Rightarrow S\backslash NP$
  - Extracting rule instances from corpus (Hockenmaier, 2003; Fowler and Penn, 2010)

Outline

- Introduction
- CCG
- Our Shift-Reduce Parser
- Experiments

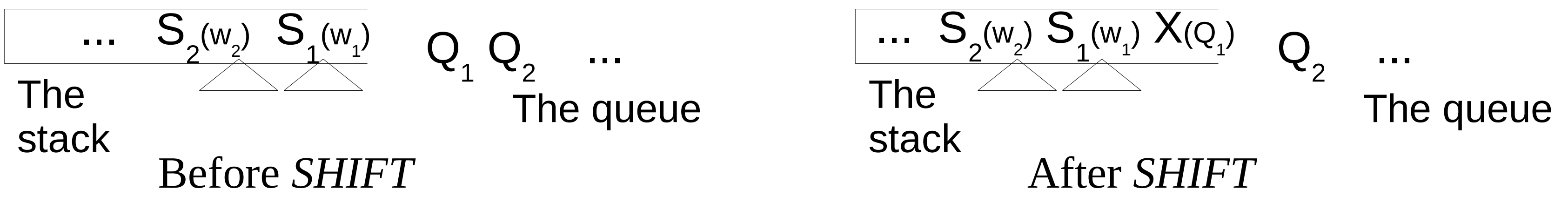
The Shift-Reduce Parser

- The shift-reduce parser
  - A *stack* of partial derivations
  - A *queue* of input words
  - A set of shift-reduce *actions* to build parsers
    - *SHIFT*
    - *COMBINE*
    - *UNARY*
    - *FINISH*



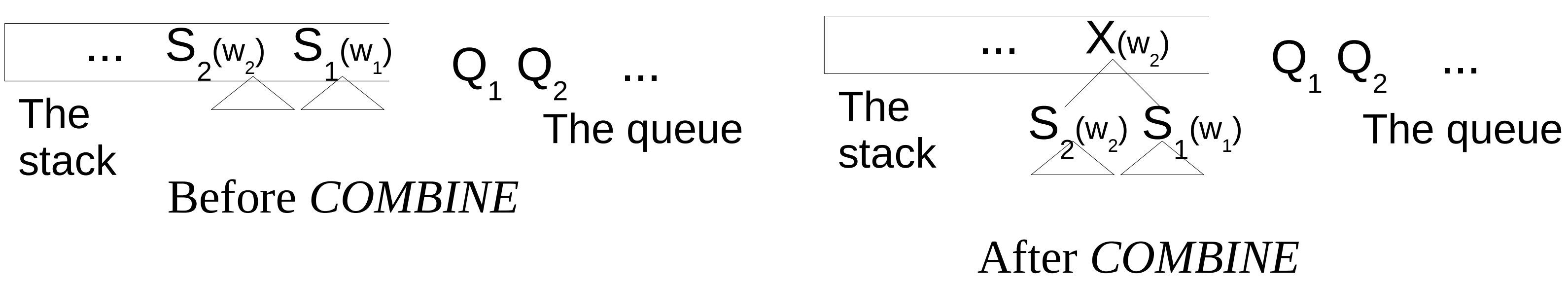
The Shift-Reduce Parser

- Shift-reduce actions
  - *SHIFT-X*
    - Pushes the head of the queue onto the stack
    - Assigns label *X* (a lexical category)
    - *SHIFT* action performs lexical category disambiguation



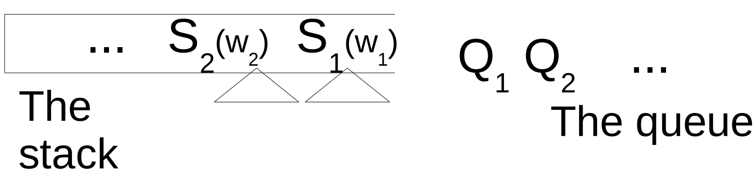
The Shift-Reduce Parser

- Shift-reduce actions
  - *COMBINE-X*
    - Pops the top two nodes off the stack
    - Combines into a new node *X*, and push it onto stack
    - Corresponds to the use of a combinatory rule in CCG

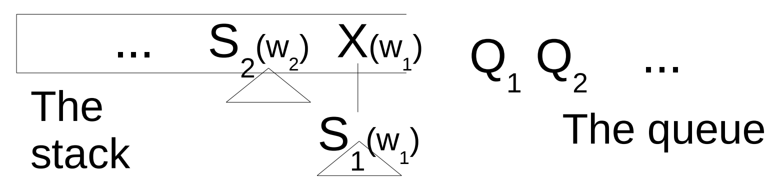


The Shift-Reduce Parser

- Shift-reduce actions
  - *UNARY-X*
    - Pops the top of the stack
    - Create a new node with category *X*; pushes it onto stack
    - Corresponds to the use of a unary rule in CCG



Before *UNARY*



After *UNARY*

The Shift-Reduce Parser

- Shift-reduce actions
  - *FINISH*
    - Terminates the parsing process
    - Can be applied when all input words have been pushed onto the stack
    - Allows fragmentary analysis:
      - when the stack holds multiple items that cannot be combined
      - such cases can arise from incorrect lexical category assignment

The Shift-Reduce Parser

- An example parsing process

IBM bought Lotus yesterday

initial

The Shift-Reduce Parser

- An example parsing process

NP_{IBM}	bought Lotus yesterday

SHIFT

The Shift-Reduce Parser

- An example parsing process

NP_{IBM} $((S[dcl]\backslash NP)/NP)_{bought}$

Lotus yesterday

SHIFT

The Shift-Reduce Parser

- An example parsing process

NP_{IBM} $((S[dcl]\backslash NP)/NP)_{bought}$ NP_{Lotus}

yesterday

SHIFT

The Shift-Reduce Parser

- An example parsing process

NP_{IBM} $(S[dcl]\backslash NP)_{bought}$

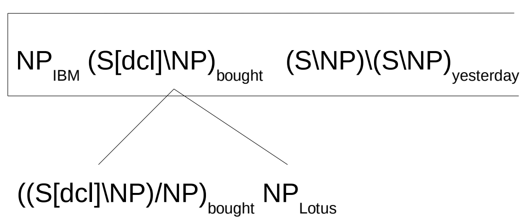
$((S[dcl]\backslash NP)/NP)_{bought}$ NP_{Lotus}

yesterday

COMBINE

The Shift-Reduce Parser

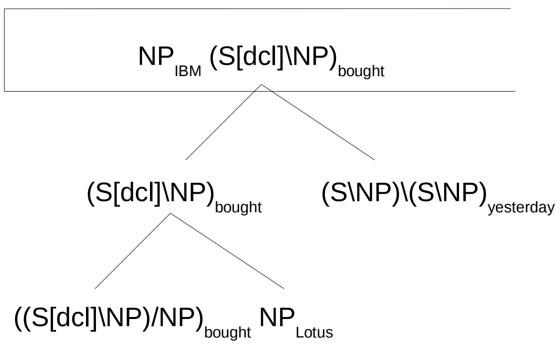
- An example parsing process



SHIFT

The Shift-Reduce Parser

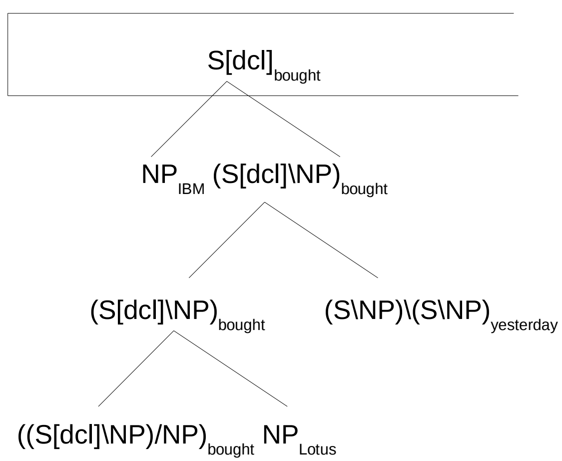
- An example parsing process



COMBINE

The Shift-Reduce Parser

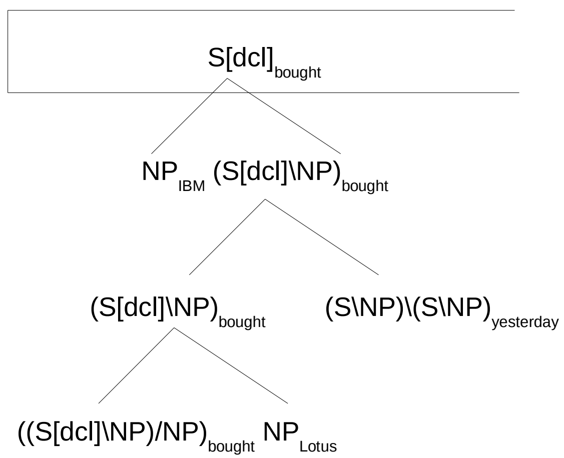
- An example parsing process



COMBINE

The Shift-Reduce Parser

- An example parsing process



FINISH

Decoding

- Candidate item <*S*, *Q*, *F*>
  - *S* – stack
  - *Q* – queue
  - *F* – boolean: parsing finished (finish action applied)
  - Start item: <*empty*, *input*, *false*>
- A derivation is built:
  - from the start item
  - by repeated application of actions
  - until the item is finished

Decoding

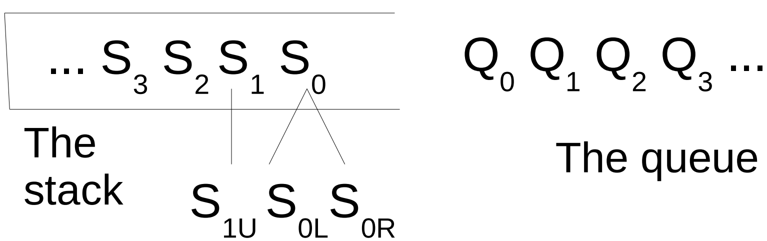
- Beam-search decoding
  - An *agenda* holds *N*-best unfinished items at each step (*N*=16)
  - A *candidate output* records the best finished item currently found
  - Initially the agenda contains only the start item
  - At each step
    - Each item in the agenda is expanded with all actions
    - Agenda replaced with *N*-best new unfinished items
    - Candidate output is updated with new finished items
  - Repeat until the agenda is empty
  - Candidate output is taken as the final output

Model and Training

- Global linear model
- Trained with:
  - the averaged perceptron (Collins, 2002)
  - "early-update" strategy (Collins and Roark, 2004)
- Zhang and Clark (2008), Zhang and Clark (2009), Huang et al., (2009)

Features

context



- Stack nodes: $S_0\ S_1\ S_2\ S_3$
- Queue nodes: $Q_0\ Q_1\ Q_2\ Q_3$
- Stack subnodes: $S_{0L}\ S_{0R}\ S_{0U}\ S_{1L/R/U}$

S0wp, S0c, S0pc, S0wc,
S1wp, S1c, S1pc, S1wc,
S2pc, S2wc,
S3pc, S3wc,

Q0wp, Q1wp, Q2wp, Q3wp,

S0Lpc, S0Lwc, S0Rpc, S0Rwc,
S0Upc, S0Uwc,
S1Lpc, S1Lwc, S1Rpc, S1Rwc,
S1Upc, S1Uwc,

S0wcS1wc, S0cS1w, S0wS1c, S0cS1c,
S0wcQ0wp, S0cQ0wp, S0wcQ0p, S0cQ0p,
S1wcQ0wp, S1cQ0wp, S1wcQ0p, S1cQ0p,

S0wcS1cQ0p, S0cS1wcQ0p, S0cS1cQ0wp,
S0cS1cQ0p, S0pS1pQ0p,
S0wcQ0pQ1p, S0cQ0wpQ1p, S0cQ0pQ1wp,
S0cQ0pQ1p, S0pQ0pQ1p,
S0wcS1cS2c, S0cS1wcS2c, S0cS1cS2wc,
S0cS1cS2c, S0pS1pS2p,

S0cS0HcS0Lc, S0cS0HcS0Rc,
S1cS1HcS1Rc,
S0cS0RcQ0p, S0cS0RcQ0w,
S0cS0LcS1c, S0cS0LcS1w,
S0cS1cS1Rc, S0wS1cS1Rc.

Outline

- Introduction
- CCG
- Our Shift-Reduce Parser
- Experiments

Experimental Data

- CCGBank (Hockenmaier and Steedman, 2007)
- Split into three subsets:
  - Training (section 02 – 21)
  - Development (section 00)
  - Testing (section 23)
- Extract CCG rules
  - Binary instances: 3070
  - Unary instances: 191

Evaluation

- Standard evaluation: F-scores over labeled CCG dependencies:
  - use evaluate script from C&C tools
- Our parser produces CCG derivation trees:
  - use generate script from C&C tools to transform derivations into dependencies

POS-tagging and Supertagging

- Supertagging is the process of lexical category assignment
- Use pos and msuper from C&C tools
- 10-fold cross-validation for training data POS-tagging and supertagging
- Multiple supertags (lexical categories) per word
  - Use a probability threshold: β
  - A small β value of 0.0001 leads to comparatively more supertags per word
  - Average 5.4 per word on the development data

Development Test Accuracies

- Labeled precision (lp.), recall (lr.) and F-score (lf.)
- Complete sentence matches (lsent.)
- Lexical category accuracy (cats.)
- Measured on:
  - all input sentences
  - C&C coverage

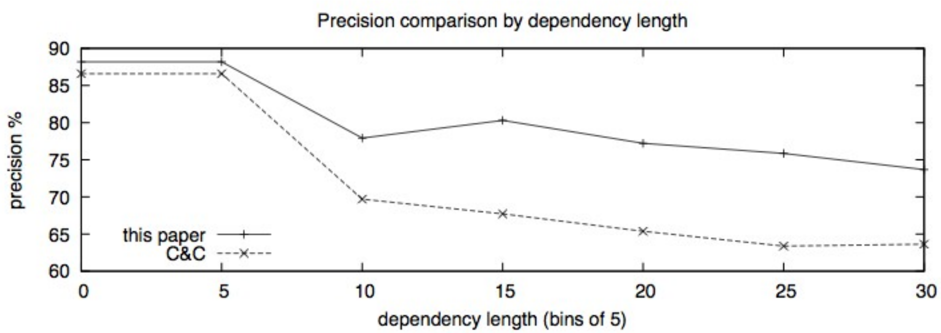
	lp.	lr.	lf.	lsent.	cats.	evaluated on
shift-reduce	87.15%	82.95%	85.00%	33.82%	92.77%	all sentences
C&C (normal-form)	85.22%	82.52%	83.85%	31.63%	92.40%	all sentences
shift-reduce	87.55%	83.63%	85.54%	34.14%	93.11%	99.06% (C&C coverage)
C&C (hybrid)	--	--	85.25%	--	--	99.06% (C&C coverage)
C&C (normal-form)	85.22%	84.29%	84.76%	31.93%	92.83%	99.06% (C&C coverage)

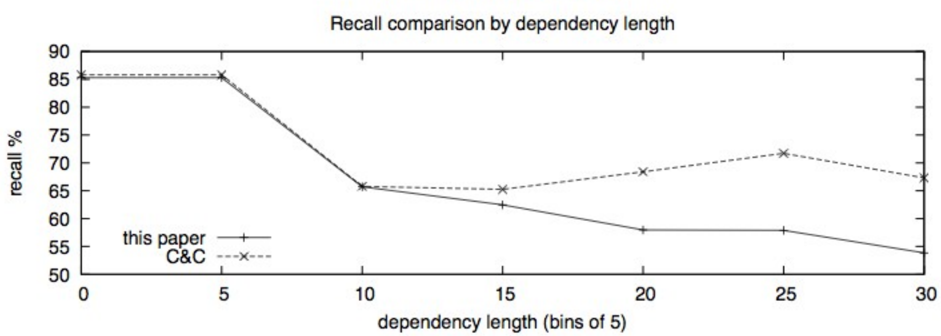
Error Comparisons

- Two different solutions to the same problem
  - Approximate beam-search with rich features
  - Optimal dynamic-programming with restricted features
- Similar to comparison between shift-reduce based MaltParser (Nivre et al., 2006) and chart-based MSTParser (McDonald et al., 2006)
- Characterize errors by sentence size, dependency length and type

Error Comparisons

- As sentence length increases
  - Both parsers give lower performance
  - No difference in the rate of accuracy degradation
- When dependency length increases





Error Comparisons

- On dependency types

category	arg	lp. (ours)	lp. (C&C)	lr. (ours)	lr. (C&C)	lf. (ours)	lf. (C&C)	freq.
N/N	1	**95.77**	95.28	**95.79**	95.62	**95.78**	95.45	7288
NP/N	1	**96.70**	96.57	**96.59**	96.03	**96.65**	96.30	4101
(NP\NP)/NP	2	**83.19**	82.17	**89.24**	88.90	**86.11**	85.40	2379
(NP\NP)/NP	1	**82.53**	81.58	**87.99**	85.74	**85.17**	83.61	2174
((S\NP)\(S\NP))/NP	3	**77.60**	71.94	71.58	**73.32**	**74.47**	72.63	1147
((S\NP)\(S\NP))/NP	2	**76.30**	70.92	70.60	**71.93**	**73.34**	71.42	1058
((S[dcl]\NP)/NP	2	**85.60**	81.57	84.30	**86.37**	**84.95**	83.90	917
PP/NP	1	73.76	**75.06**	**72.83**	70.09	**73.29**	72.49	876
((S[dcl]\NP)/NP	1	**85.32**	81.62	82.00	**85.55**	**83.63**	83.54	872
((S\NP)\(S\NP))	2	84.44	**86.85**	86.60	**86.73**	85.51	**86.79**	746

Test results

- F&P = Fowler and Penn (2010)

	lp.	lr.	lf.	lsent.	cats.	evaluated
shift-reduce	87.43	83.61	85.48	35.19	93.12	all sentences
C&C (normal-form)	85.58	82.85	84.20	32.90	92.84	all sentences
shift-reduce	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)
F&P (Petrov I-5)*	86.29	85.73	86.01	--	--	-- (F&P ∩ C&C coverage; 96.65% on dev. test)
C&C hybrid*	86.46	85.11	85.78	--	--	-- (F&P ∩ C&C coverage; 96.65% on dev. test)

Conclusions

- First work to show competitive results for shift-reduce CCG parsing
- Comparison between shift-reduce parsing and chart parsing for CCG
- Complementary errors made by our parser and C&C suggests the possibility of parser combination

Thank you!

(thanks to Michael Auli and Joakim Nivre for assistance)