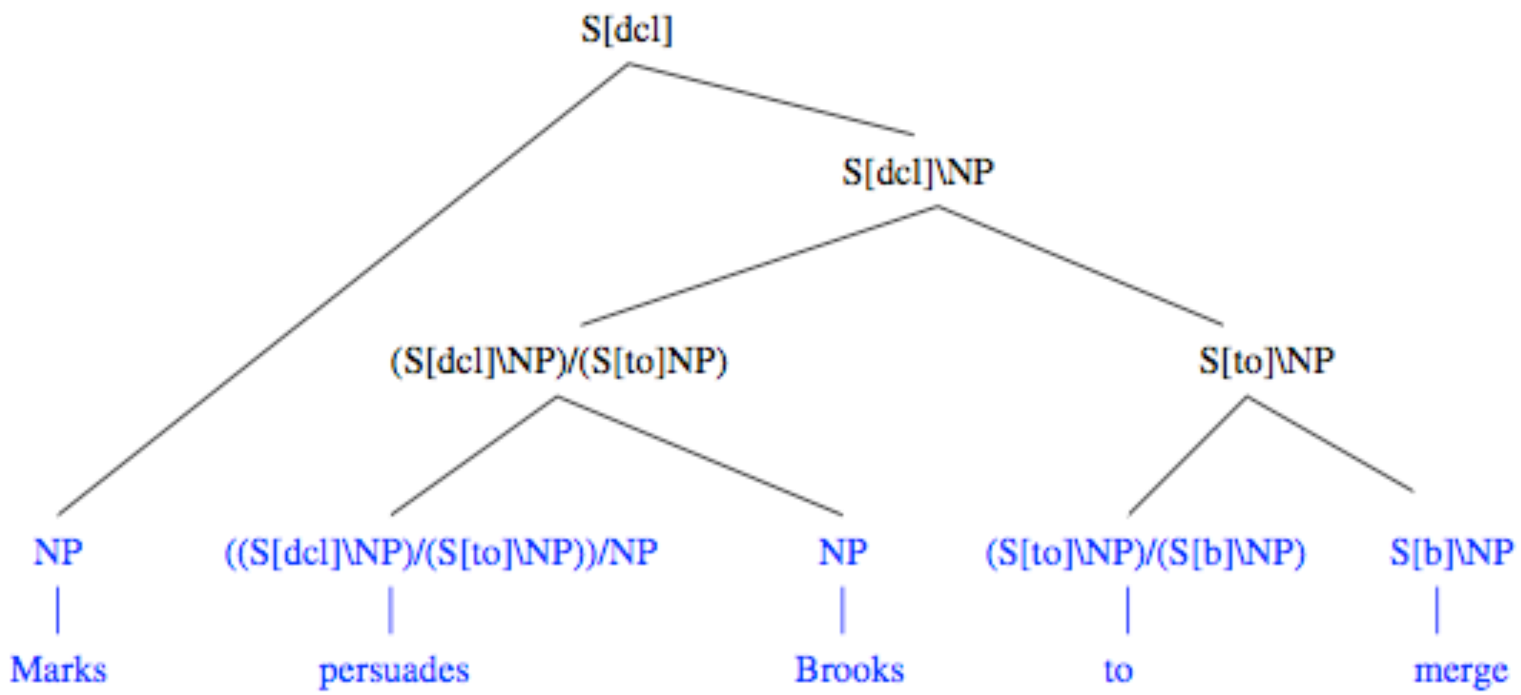
Introduction to Syntax and Parsing
ACS 2015/16
Stephen Clark
L8: Parsing with CCG

UNIVERSITY OF CAMBRIDGE

Inducing a Grammar from CCGbank



- Grammar (lexicon) can be read off the leaves of the trees

UNIVERSITY OF CAMBRIDGE

Chart Parsing with CCG

- Stage 1
  - Assign POS tags and lexical categories to words in the sentence
  - Use taggers to assign the POS tags and categories
    – based on standard Maximum Entropy tagging techniques
- Stage 2
  - Combine the categories using the combinatory rules
  - Can use standard bottom-up CKY chart-parsing algorithm
- Stage 3
  - Find the highest scoring derivation according to some model
    – e.g. generative model, CRF, perceptron
  - Viterbi algorithm finds this efficently

UNIVERSITY OF CAMBRIDGE

CCG Supertagging

He	goes	on	the	road	with	his	piano
NP	$(S[dcl]\backslash NP)/PP$	PP/NP	NP/N	N	$((S\backslash NP)\backslash(S\backslash NP))/NP$	NP/N	N

A	bitter	conflict	with	global	implications
NP/N	N/N	N	$(NP\backslash NP)/NP$	N/N	N

- Baseline tagging accuracy is $\approx$ 72%
  - baseline is to assign tag most frequently seen with word in training data, and assign N to unseen words
- Baseline for Penn Treebank POS tagging is $\approx$ 90%

CCG Multitagging

- Per-word tagging accuracy is $\approx 92\%$
- Potentially assign more than one category to a word
  - assign all categories whose probability is within some factor β of the highest probability category
- Accuracy is over 97% at only 1.4 categories per word
- Accuracy is now high enough to serve as a front-end to the parser

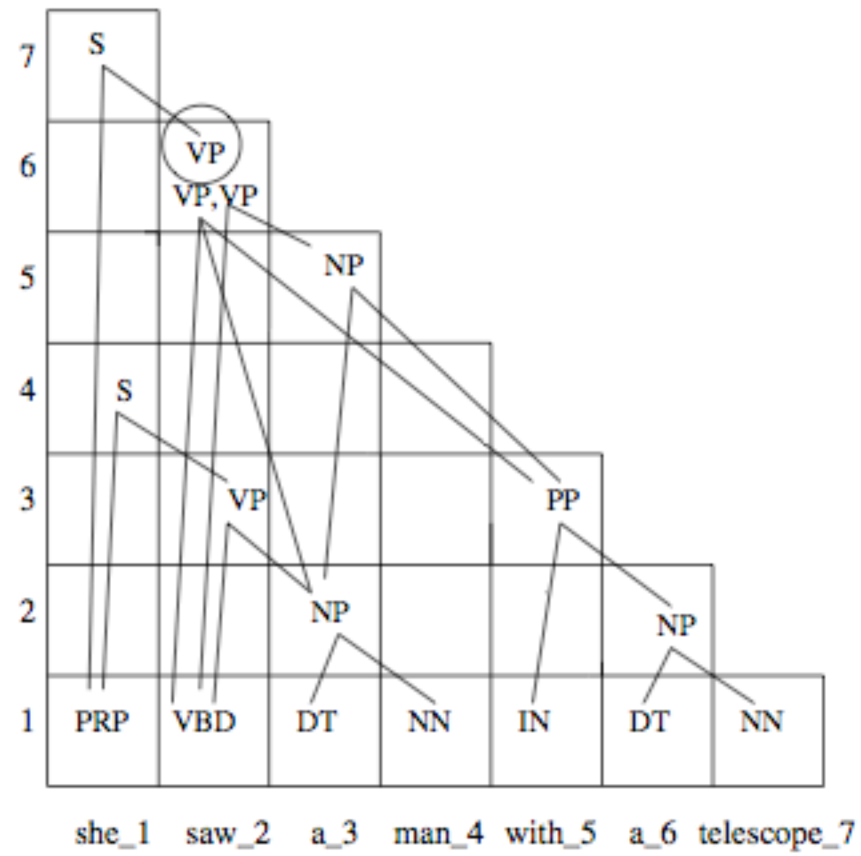
CKY Algorithm

```
chart[i][j] is a cell containing categories spanning words from i to i + j

initialise chart with categories of span 1 (lexical categories)

LOOP over span of result category (j = 2 to SENT_LENGTH)
 LOOP over start position of left combining category (i = 0 to SENT_LENGTH - j)
  LOOP over span of left combining category (k = 1 to j - 1)
   chart[i][j] ++ Combine(chart[i][k], chart[i + k][j - k])
```

Chart Parsing



- DP algorithms can be run over the packed representation
- The *Viterbi* algorithm finds the highest scoring derivation

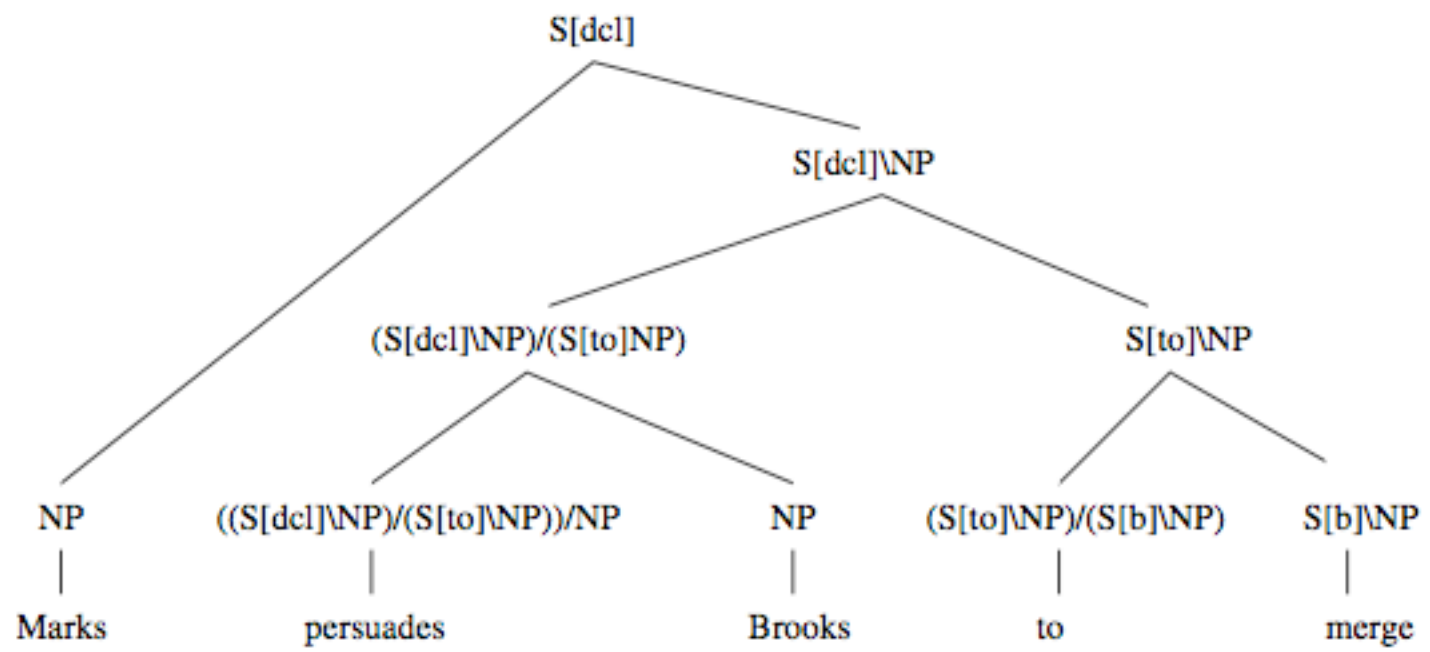
Linear Parsing Model

$$\text{Score}(d, S) = \sum_i \lambda_i f_i(d) = \overline{\lambda} \cdot \phi(d)$$

- Features are counts over d
  - root category of d (plus lexical head)
  - $\langle$lexical category, lexical item$\rangle$ pairs
  - rule feature: $S \rightarrow NP \quad S\backslash NP$ (plus lexical head)
  - predicate argument dependency: $\text{subj}(\text{bought}, \text{IBM})$ (plus distance)
  - "Backing-off" features with words replaced by POS tags
- Use Perceptron training to set the weights

UNIVERSITY OF CAMBRIDGE

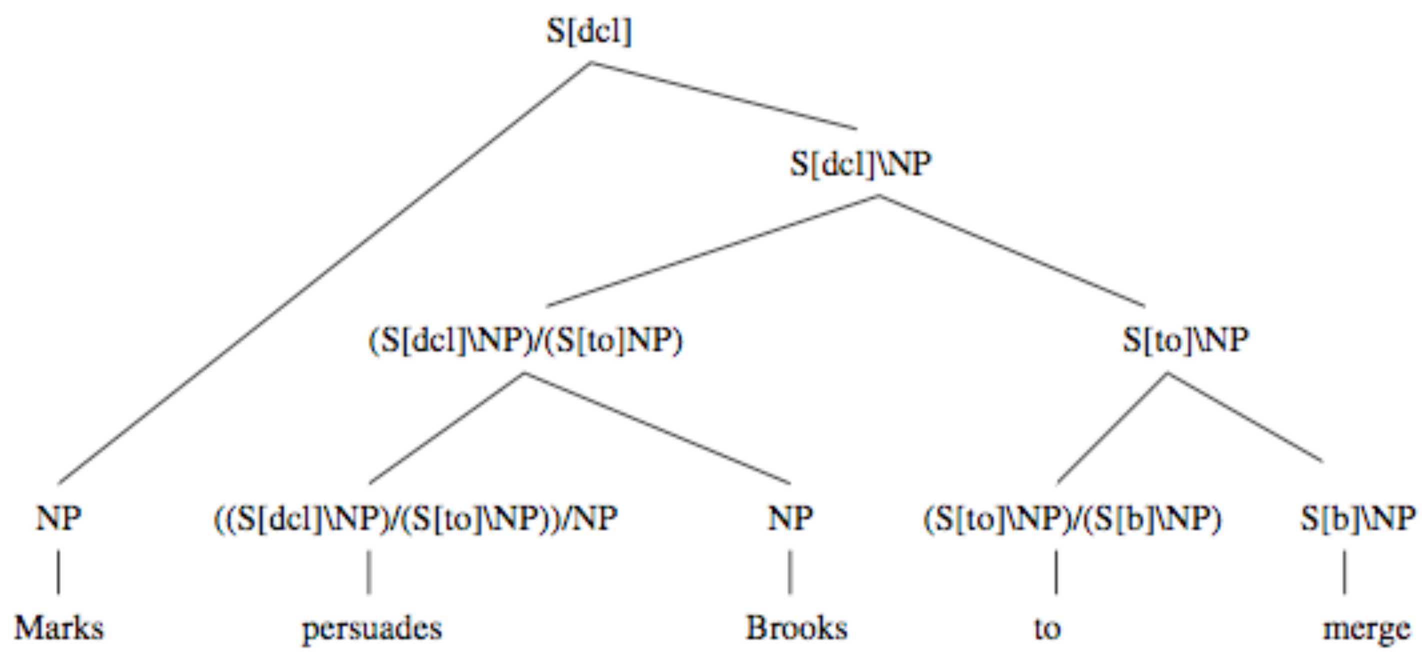
Training Data from CCGbank



subj(persuades, Marks)
obj(persuades, Brooks)
subj(merge, Brooks)
to-inf(persuades, merge)

UNIVERSITY OF CAMBRIDGE

Feature Representation



$$f_i : D \rightarrow \mathcal{N} \qquad (3\,000\,000 \leq i \leq 1)$$

Linear Parsing Model

$$\mathsf{Score}(d, s) = \sum_i \lambda_i . f_i(d) = \overline{\lambda} \cdot \overline{f}(d)$$

- f_i are the *features* (defined by hand)
- λ_i are the corresponding *weights* (which need to be learned)

UNIVERSITY OF CAMBRIDGE

Perceptron Training

$$\text{Score}(d, S) = \sum_i \lambda_i f_i(d) = \bar{\lambda} \cdot \phi(d)$$

Inputs: training examples (x_i, y_i)
Initialisation: set $\bar{\lambda} = 0$
Algorithm:
for $t = 1..T$, $i = 1..N$
calculate $z_i = \arg\max_{y \in \text{GEN}(x_i)} \Phi(x_i, y) \cdot \bar{\lambda}$
if $z_i \neq y_i$
$\bar{\lambda} = \bar{\lambda} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
Outputs: $\bar{\lambda}$

Perceptron Training

W0 = <0,0,0,....,0,0,....,0,...0,0,0,0,....,0>

5	S S\NP S/S				
4	S/S S/NP	S\NP S\S (S\NP)\NP			
3	S	S\NP S/NP	NP VP\VP		
2	S/NP S/S	S\NP (S\NP)/PP	PP NP\NP	NP\NP VP\VP	
1	NP N S/(S\NP) (S/S)/NP	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP	NP N
SENT1:	w1	w2	w3	w4	w5

Perceptron Training

DECODE:

5	S S\NP S/S				
4	S/S S/NP	S\NP S\S (S\NP)\NP			
3	S	S\NP S/NP	NP VP\VP		
2	S/NP S/S	S\NP (S\NP)/PP	PP NP\NP	NP\NP VP\VP	
1	NP N S/(S\NP) (S/S)/NP	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP	NP N
SENT1:	w1	w2	w3	w4	w5

W0 = <0,0,0,...,0,0,...,0,...0,0,0,0,...,0>

f1, f20, f55, f100, f210, f345

f19, f25, f75, f150, f211, f346, f450, f500, f525

f15, f21, f56, f120, f212, f348, f419

Perceptron Training

UPDATE WEIGHTS:

5	S S\NP S/S				
4	S/S S/NP	S\NP S\S (S\NP)\NP			
3	S	S\NP S/NP	NP VP\VP		
2	S/NP S/S	S\NP (S\NP)/PP	PP NP\NP	NP\NP VP\VP	
1	NP N S/(S\NP) (S/S)/NP	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP	NP N
SENT1:	w1	w2	w3	w4	w5

W1 = <0,1,0,....,-1,0,....,-1,...0,1,0,-1,....,0>

f1, f20, f55, f100, f210, f345
f19, f25, f75, f150, f211, f346, f450, f500, f525
f15, f21, f56, f120, f212, f348, f419

Perceptron Training

W1 = <0,1,0,....,-1,0,....,-1,....0,1,0,-1,....,0>

4	S/S S/NP S			
3	S	S\NP PP/NP		
2	S/NP S/S	S\NP (S\NP)/PP	PP PP/NP NP\NP	
1	NP N S/(S\NP)	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP NP
SENT2:	w1	w2	w3	w4

Perceptron Training

W1 = <0,1,0,...,-1,0,...,-1,...0,1,0,-1,...,0>

DECODE:

4	S/S S/NP S			
3	S	S\NP PP/NP		
2	S/NP S/S	S\NP (S\NP)/PP	PP PP/NP NP\NP	
1	NP N S/(S\NP)	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP NP
SENT2:	w1	w2	w3	w4

f11, f21, f57, f90, f145, f250

f21, f25, f76, f151, f222, f348, f444, f507, f575

f17, f45, f155, f167, f678

UNIVERSITY OF CAMBRIDGE

Perceptron Training

W2 = <0,2,-1,....,-1,1,....,-1,...0,1,0,-2,....,-1>

f11, f21, f57, f90, f145, f250
f21, f25, f76, f151, f222, f348, f444, f507, f575
f17, f45, f155, f167, f678

UPDATE WEIGHTS:

4	S/S S/NP S			
3	S	S\NP PP/NP		
2	S/NP S/S	S\NP (S\NP)/PP	PP PP/NP NP\NP	
1	NP N S/(S\NP)	(S\NP)/NP S\NP (S\NP)/PP	NP N PP/PP	(NP\NP)/NP (VP\VP)/NP PP NP
SENT2:	w1	w2	w3	w4

UNIVERSITY OF CAMBRIDGE

DP vs. Beam Search

- DP requires the optimal sub-problem property
- For efficient parsing this restricts the feature set
- An alternative is to apply a beam to each cell
- Now no restrictions on the features
- Max-violation perceptron used for training

Parser Evaluation

- Compare output of the parser with a *gold standard*
- Exact match metric sometimes used but a little crude
- Partial match against a set of *grammatical relations* currently the method of choice
  - measures recovery of semantically important relations
  - relatively theory-neutral representation

Head-based GRs

- *She gave the present to Kim*
  ```
  (ncsubj gave She _)
  (dobj gave present)
  (iobj gave to)
  (dobj to Kim)
  (det present the)
  ```
- *The company wants to wean itself away from expensive gimmicks*
  ```
  (xcomp to wants wean)
  (iobj wean from)
  (ncmod prt wean away)
  (dobj wean itself)
  (dobj from gimmicks)
  (ncmod _ gimmicks expensive)
  ...
  ```

UNIVERSITY OF CAMBRIDGE

Mapping CCG Dependencies to GRs

- Argument slots in CCG dependencies are mapped to GRs

CCG lexical category	arg slot	GR
$(S[dcl]\backslash NP_1)/NP_2$	1	(nsubj %l %f)
$(S[dcl]\backslash NP_1)/NP_2$	2	(dobj %l %f)
$(NP\backslash NP_1)/NP_2$	1	(prep %f %l)
$(NP\backslash NP_1)/NP_2$	2	(pobj %l %f)
$NP[nb]/N_1$	1	(det %f %l)

- Mapping is many-to-many

Test Suite: DepBank

- 700 sentences of newspaper text manually annotated with GRs
- Calculate precision and recall over GRs

$$\mathit{Prec} = \frac{\#\ \mathit{correct}}{\#\ \mathit{proposed\ by\ parser}} \qquad \mathit{Rec} = \frac{\#\ \mathit{correct}}{\#\ \mathit{in\ gold\ standard}}$$

$$\mathit{F\text{-}score} = \frac{2\,P\,R}{P+R}$$

UNIVERSITY OF CAMBRIDGE

Parsing Accuracy

Prec	Rec	F-score
84.1	82.8	83.4

GR	F-score
ncsubj	79.6
dobj	87.7
obj2	66.7
iobj	73.4
clausal	75.0
ncmod	76.1
aux	92.8
det	95.1
conj	77.5