Statistical NLP
Spring 2007

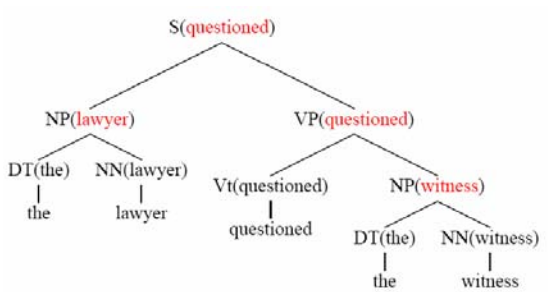
Lecture 18: Semantic Roles

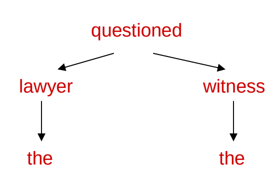
Dan Klein – UC Berkeley

Includes examples from Johnson, Jurafsky and Gildea, Palmer

Dependency Parsing

- Lexicalized parsers can be seen as producing *dependency trees*

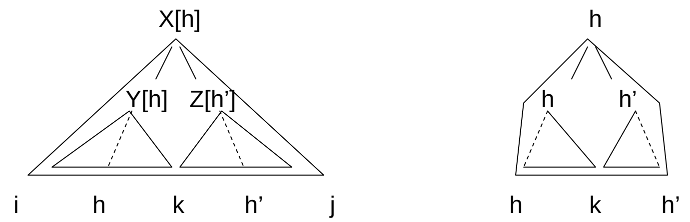




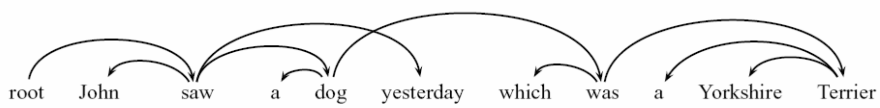
- Each local binary tree corresponds to an attachment in the dependency graph

Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]

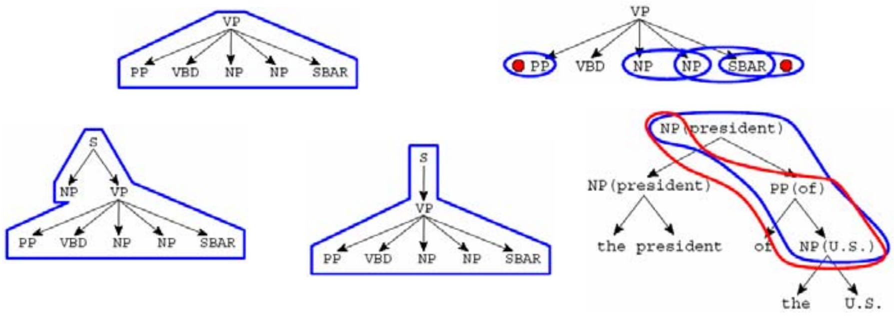


- Some work on *non-projective* dependencies
  - Common in, e.g. Czech parsing
  - Can do with MST algorithms [McDonald and Pereira 05]



Parse Reranking

- Assume the number of parses is very small
- We can represent each parse T as an arbitrary feature vector $\varphi(T)$
  - Typically, all local rules are features
  - Also non-local features, like how right-branching the overall tree is
  - [Charniak and Johnson 05] gives a rich set of features

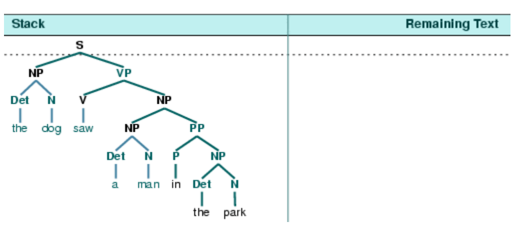


Parse Reranking

- Since the number of parses is no longer huge
  - Can enumerate all parses efficiently
  - Can use simple machine learning methods to score trees
  - E.g. maxent reranking: learn a binary classifier over trees where:
    - The top candidates are positive
    - All others are negative
    - Rank trees by P(+|T)
- The best parsing numbers are from reranking systems

Shift-Reduce Parsers

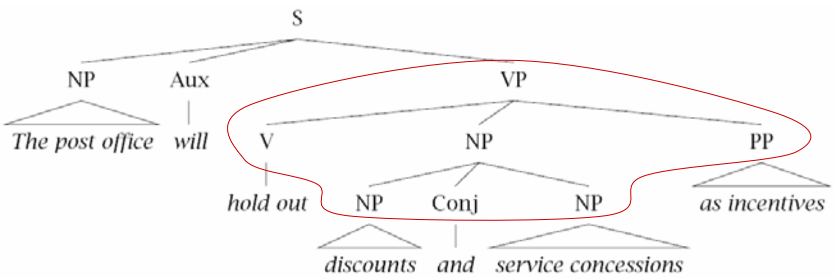
- Another way to derive a tree:



- Parsing
  - No useful dynamic programming search
  - Can still use beam search [Ratnaparkhi 97]

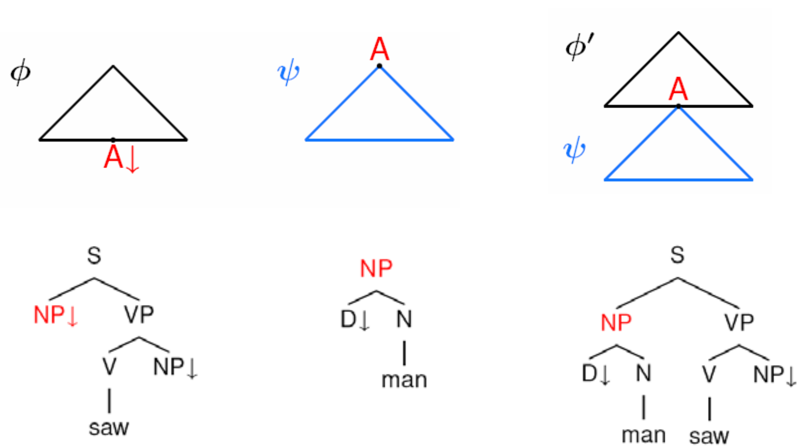
Data-oriented parsing:

- Rewrite large (possibly lexicalized) subtrees in a single step



- Formally, a *tree-insertion grammar*
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable *parse* is NP-complete

TIG: Insertion



Derivational Representations

- Generative derivational models:

$$P(D) = \prod_{d_i \in D} P(d_i | d_0 \ldots d_{i-1})$$

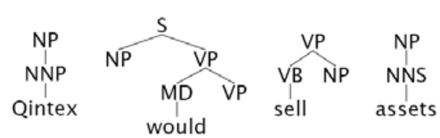
- How is a PCFG a generative derivational model?
- Distinction between *parses* and *parse derivations*.

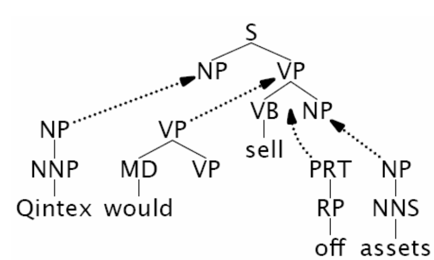
$$P(T) = \sum_{D:D \rightarrow T} P(D)$$

- How could there be multiple derivations?

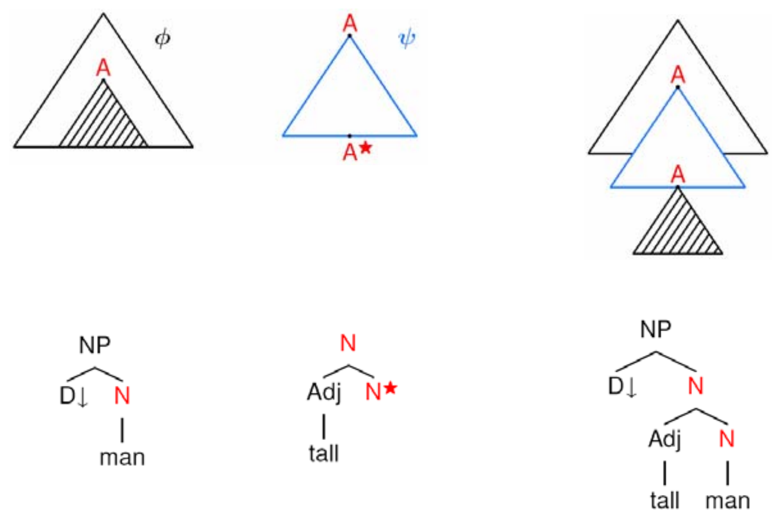
Tree-adjoining grammars

- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- … as well as other weird stuff that CFGs don't capture well (e.g. cross-serial dependencies)

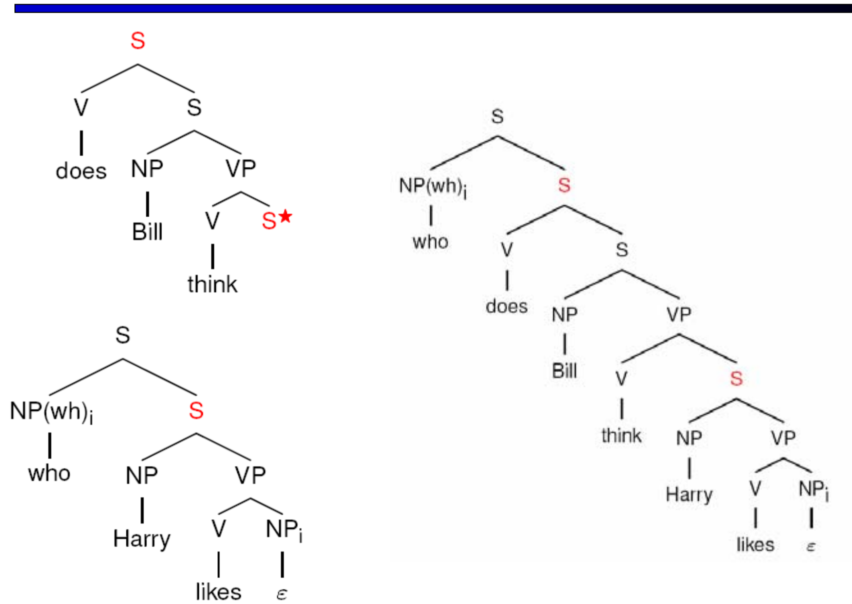




TAG: Adjunction



TAG: Long Distance



CCG Parsing

- Combinatory Categorial Grammar
  - Fully (mono-) lexicalized grammar
  - Categories encode argument sequences
  - Very closely related to the lambda calculus (more later)
  - Can have spurious ambiguities (why?)

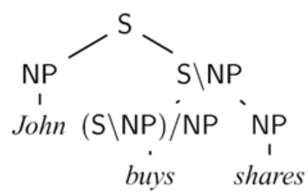
$$\textit{John} \vdash \mathrm{NP}$$

$$\textit{shares} \vdash \mathrm{NP}$$

$$\textit{buys} \vdash (\mathrm{S}\backslash\mathrm{NP})/\mathrm{NP}$$

$$\textit{sleeps} \vdash \mathrm{S}\backslash\mathrm{NP}$$

$$\textit{well} \vdash (\mathrm{S}\backslash\mathrm{NP})\backslash(\mathrm{S}\backslash\mathrm{NP})$$



Statistical Semantics?

- Last time:
  - Syntactic trees + lexical translations → (unambiguous) logical statements
  - Symbolic deep (?) semantics
  - Often used as part of a logical NLP interface or in database / dialog systems
  - Applications like question answering and information extraction often don't need such expressiveness

- Today:
  - Statistically extracting shallow semantics
  - Semantic role labeling
  - Coreference resolution

Semantic Role Labeling (SRL)

- Characterize clauses as *relations* with *roles*:

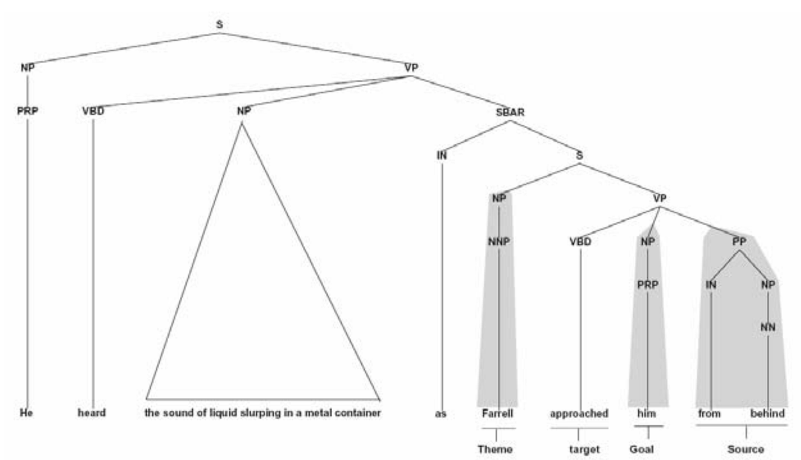
[$_{Judge}$ She] **blames** [$_{Evaluee}$ the Government] [$_{Reason}$ for failing to do enough to help] .

Holman would characterise this as **blaming** [$_{Evaluee}$ the poor] .

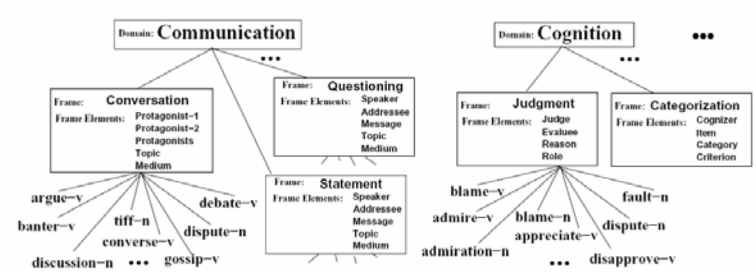
The letter quotes Black as saying that [$_{Judge}$ white and Navajo ranchers] misrepresent their livestock losses and **blame** [$_{Reason}$ everything] [$_{Evaluee}$ on coyotes] .

- Want to more than which NP is the subject (but not much more):
- Relations like *subject* are syntactic, relations like *agent* or *message* are semantic
- Typical pipeline:
  - Parse, then label roles
  - Almost all errors locked in by parser
  - Really, SRL is quite a lot easier than parsing

SRL Example



PropBank / FrameNet



- FrameNet: roles shared between verbs
- PropBank: each verb has it's own roles
- PropBank more used, because it's layered over the treebank (and so has greater coverage, plus parses)
- Note: some linguistic theories postulate even fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)

PropBank Example

fall.01 sense: move downward

roles:	Arg1:	thing falling
	Arg2:	extent, distance fallen
	Arg3:	start point
	Arg4:	end point

Sales fell to $251.2 million from $278.7 million.

arg1:	Sales
rel:	fell
arg4:	to $251.2 million
arg3:	from $278.7 million

PropBank Example

rotate.02 sense: shift from one thing to another
roles: Arg0: causer of shift
Arg1: thing being changed
Arg2: old thing
Arg3: new thing

Many of Wednesday's winners were losers yesterday as investors quickly took profits and rotated their buying to other issues, traders said. (wsj_1723)

arg0: investors
rel: rotated
arg1: their buying
arg3: to other issues

PropBank Example

aim.01 sense: intend, plan
roles: Arg0: aimer, planner
Arg1: plan, intent

The Central Council of Church Bell Ringers aims *trace* to improve relations with vicars. (wsj_0089)

arg0: The Central Council of Church Bell Ringers
rel: aims
arg1: *trace* to improve relations with vicars

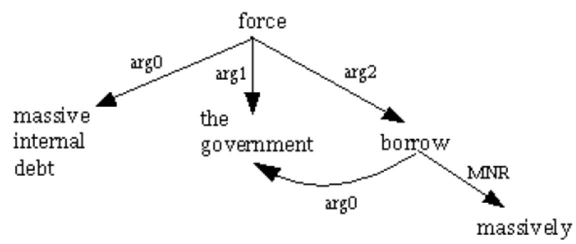
aim.02 sense: point (weapon) at
roles: Arg0: aimer
Arg1: weapon, etc.
Arg2: target

Banks have been aiming packages at the elderly.

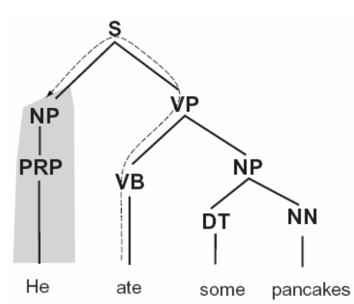
arg0: Banks
rel: aiming
arg1: packages
arg2: at the elderly

Shared Arguments

(NP-SBJ (JJ massive) (JJ internal) (NN debt))
(VP (VBZ has)
(VP (VBN forced)
(S
(NP-SBJ-1 (DT the) (NN government))
(VP
(VP (TO to)
(VP (VB borrow)
(ADVP-MNR (RB massively))...



Path Features



Path	*Description*
VB↑VP↓PP	PP argument/adjunct
VB↑VP↑S↓NP	subject
VB↑VP↓NP	object
VB↑VP↑VP↑S↓NP	subject (embedded VP)
VB↑VP↓ADVP	adverbial adjunct
NN↑NP↑NP↓PP	prepositional complement of noun

Results

- Features:
  - Path from target to filler
  - Filler’s syntactic type, headword, case
  - Target’s identity
  - Sentence voice, etc.
  - Lots of other second-order features

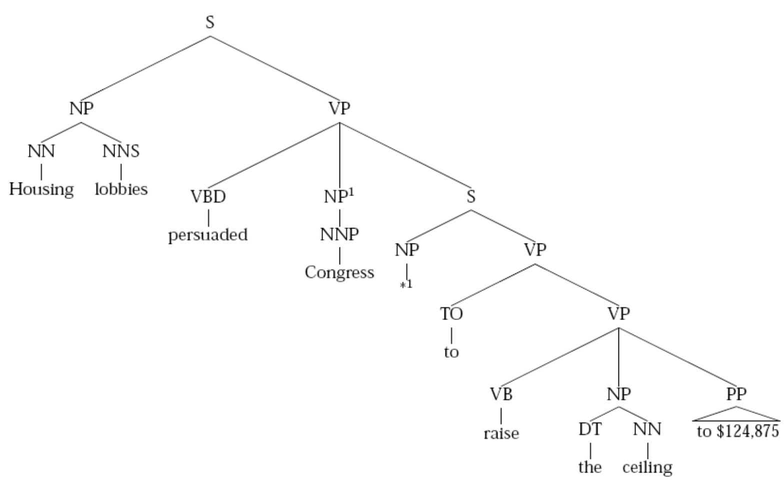
- Gold vs parsed source trees
  - SRL is fairly easy on gold trees

CORE		ARGM	
F1	Acc.	F1	Acc.
92.2	80.7	89.9	71.8

  - Harder on automatic parses

CORE		ARGM	
F1	Acc.	F1	Acc.
84.1	66.5	81.4	55.6

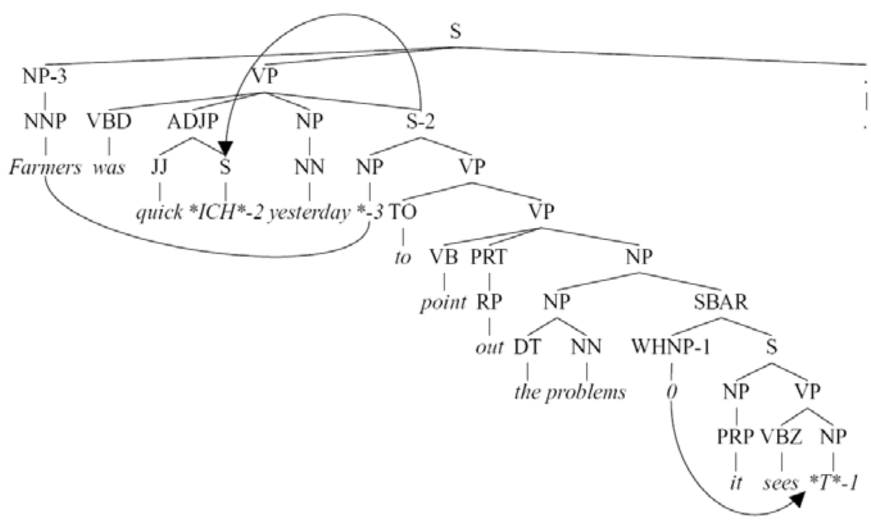
Interaction with Empty Elements



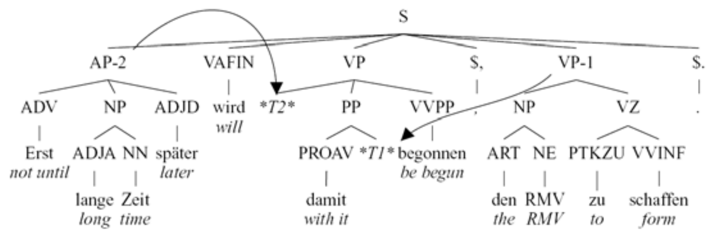
Empty Elements

- In the PTB, three kinds of empty elements:
  - Null items (usually complementizers)
  - Dislocation (WH-traces, topicalization, relative clause and heavy NP extraposition)
  - Control (raising, passives, control, shared argumentation)
- Need to reconstruct these (and resolve any indexation)

Example: English

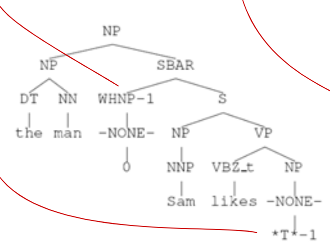


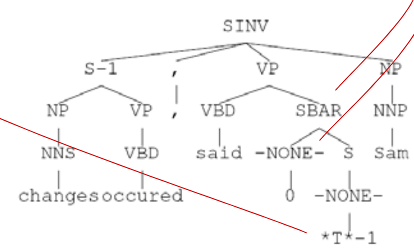
Example: German



Types of Empties

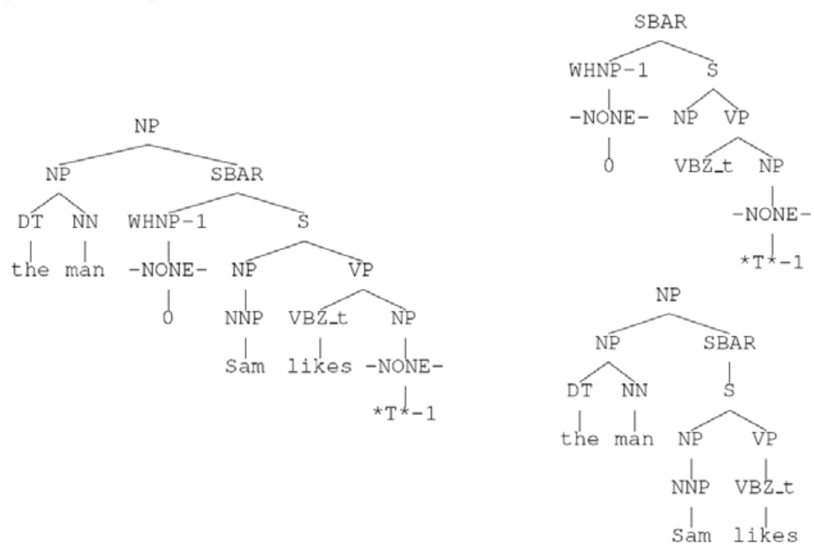
Antecedent	POS	Label	Count	Description
NP	NP	*	18,334	NP trace (e.g., Sam was seen *)
	NP	*	9,812	NP PRO (e.g., * to sleep is nice)
WHNP	NP	*T*	8,620	WH trace (e.g., the woman who you saw *T*)
		U	7,478	Empty units (e.g., $ 25 *U*)
		0	5,635	Empty complementizers (e.g., Sam said 0 Sasha snores)
S	S	*T*	4,063	Moved clauses (e.g., Sam had to go, Sasha explained *T*)
WHADVP	ADVP	*T*	2,492	WH-trace (e.g., Sam explained how to leave *T*)
	SBAR		2,033	Empty clauses (e.g., Sam had to go, Sasha explained (SBAR))
	WHNP	0	1,759	Empty relative pronouns (e.g., the woman 0 we saw)
	WHADVP	0	575	Empty relative pronouns (e.g., no reason 0 to leave)





A Pattern-Matching Approach

- [Johnson 02]



Pattern-Matching Details

- Something like transformation-based learning
- Extract patterns
  - Details: transitive verb marking, auxiliaries
  - Details: legal subtrees
- Rank patterns
  - Pruning ranking: by correct / match rate
  - Application priority: by depth
- Pre-order traversal
- Greedy match

Top Patterns Extracted

Count	Match	Pattern
5816	6223	(S (NP (-NONE- *)) VP)
5605	7895	(SBAR (-NONE- 0) S)
5312	5338	(SBAR WHNP-1 (S (NP (-NONE- *T*-1)) VP))
4434	5217	(NP QP (-NONE- *U*))
1682	1682	(NP $ CD (-NONE- *U*))
1327	1593	(VP VBN_t (NP (-NONE- *)) PP)
700	700	(ADJP QP (-NONE- *U*))
662	1219	(SBAR (WHNP-1 (-NONE- 0)) (S (NP (-NONE- *T*-1)) VP))
618	635	(S S-1 , NP (VP VBD (SBAR (-NONE- 0) (S (-NONE- *T*-1)))) .)
499	512	(SINV `` S-1 , '' (VP VBZ (S (-NONE- *T*-1))) NP .)
361	369	(SINV `` S-1 , '' (VP VBD (S (-NONE- *T*-1))) NP .)
352	320	(S NP-1 (VP VBZ (S (NP (-NONE- *-1)) VP)))
346	273	(S NP-1 (VP AUX (VP VBN_t (NP (-NONE- *-1)) PP)))
322	467	(VP VBD_t (NP (-NONE- *)) PP)
269	275	(S `` S-1 , '' NP (VP VBD (S (-NONE- *T*-1))) .)

Results

Empty node		Section 23			Parser output		
POS	Label	*P*	*R*	*f*	*P*	*R*	*f*
(Overall)		0.93	0.83	0.88	0.85	0.74	0.79
NP	*	0.95	0.87	0.91	0.86	0.79	0.82
NP	*T*	0.93	0.88	0.91	0.85	0.77	0.81
	0	0.94	0.99	0.96	0.86	0.89	0.88
	U	0.92	0.98	0.95	0.87	0.96	0.92
S	*T*	0.98	0.83	0.90	0.97	0.81	0.88
ADVP	*T*	0.91	0.52	0.66	0.84	0.42	0.56
SBAR		0.90	0.63	0.74	0.88	0.58	0.70
WHNP	0	0.75	0.79	0.77	0.48	0.46	0.47

A Machine-Learning Approach

- [Levy and Manning 04]
- Build two classifiers:
  - First one predicts where empties go
  - Second one predicts if/where they are bound
  - Use syntactic features similar to SRL (paths, categories, heads, etc)

	Performance on gold trees							Performance on parsed trees					
	ID			Rel	Combo			ID			Combo		
	P	R	F1	Acc	P	R	F1	P	R	F1	P	R	F1
WSJ(full)	92.0	82.9	87.2	95.0	89.6	80.1	84.6	34.5	47.6	40.0	17.8	24.3	20.5
WSJ(sm)	92.3	79.5	85.5	93.3	90.4	77.2	83.2	38.0	47.3	42.1	19.7	24.3	21.7
NEGRA	73.9	64.6	69.0	85.1	63.3	55.4	59.1	48.3	39.7	43.6	20.9	17.2	18.9