

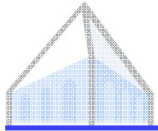
Natural Language Processing



Parsing IV

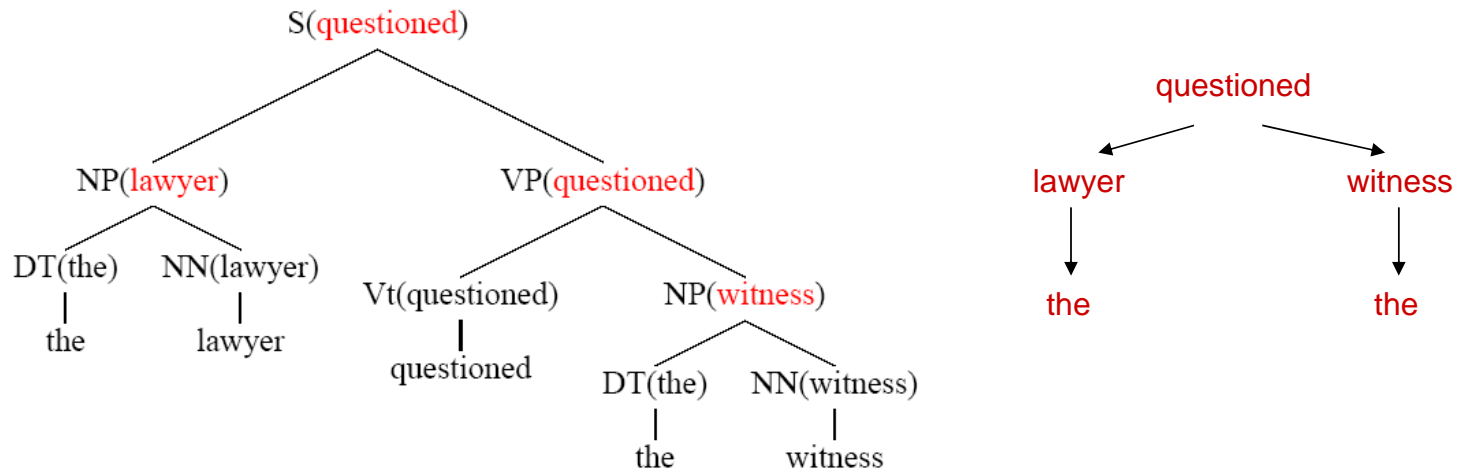
Dan Klein – UC Berkeley

Other Syntactic Models

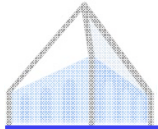


Dependency Parsing

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

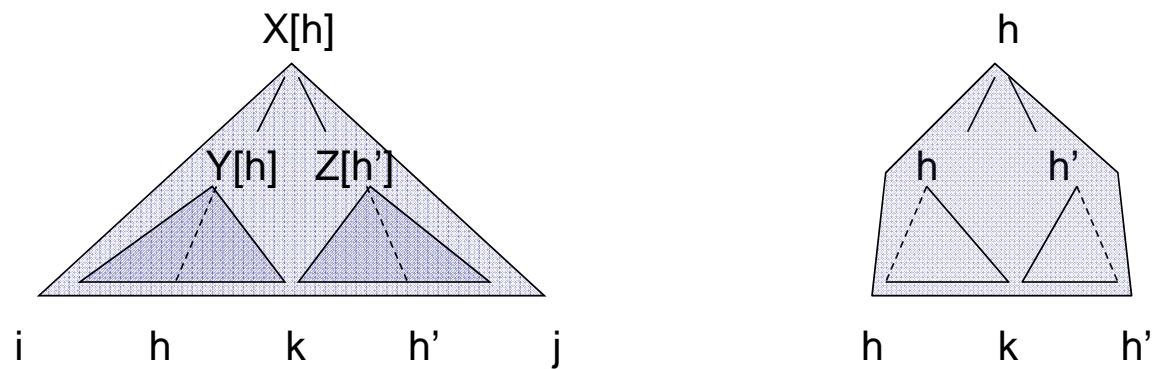


- Each local binary tree corresponds to an attachment in the dependency graph

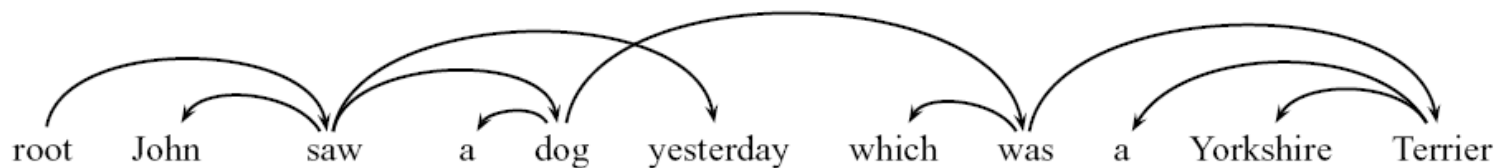


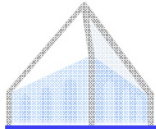
Dependency Parsing

- Pure dependency parsing is only cubic [Eisner 99]



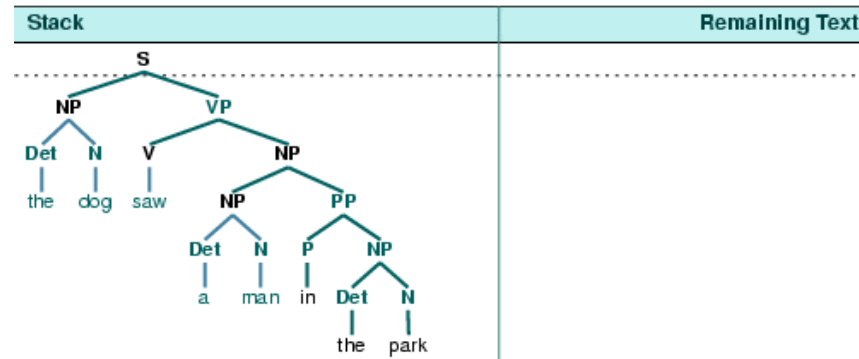
- Some work on *non-projective* dependencies
 - Common in, e.g. Czech parsing
 - Can do with MST algorithms [McDonald and Pereira 05]



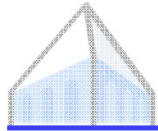


Shift-Reduce Parsers

- Another way to derive a tree:

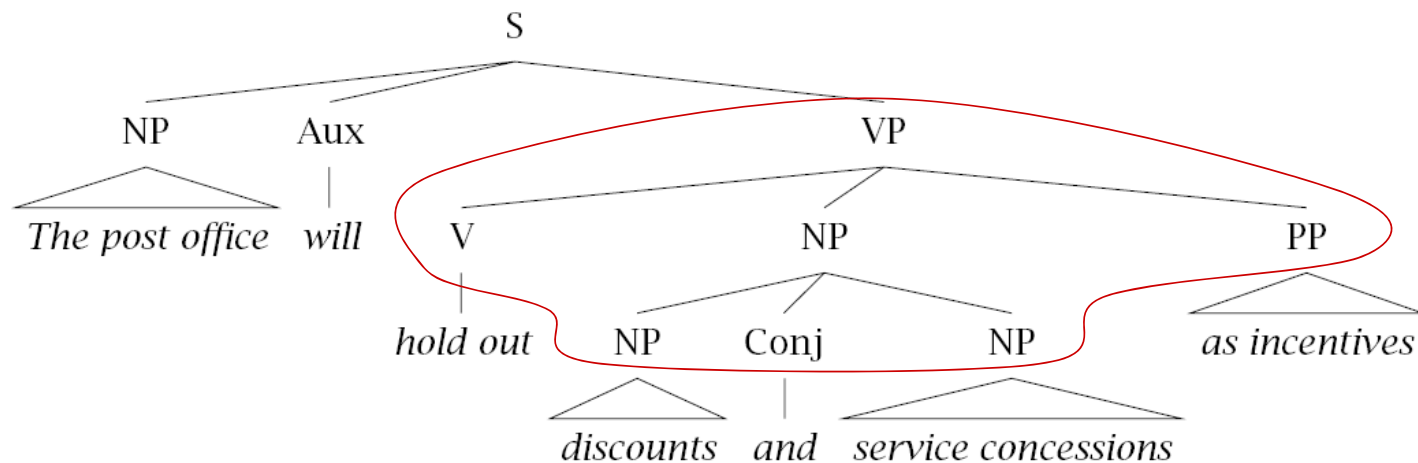


- Parsing
 - No useful dynamic programming search
 - Can still use beam search [Ratnaparkhi 97]

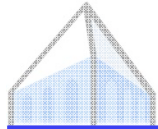


Tree Insertion Grammars

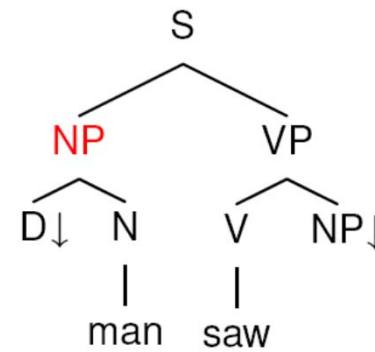
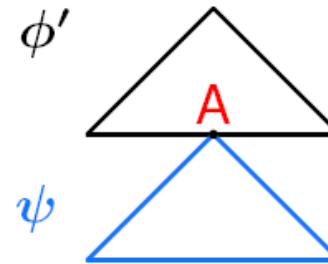
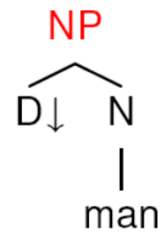
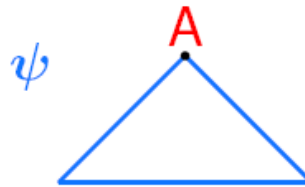
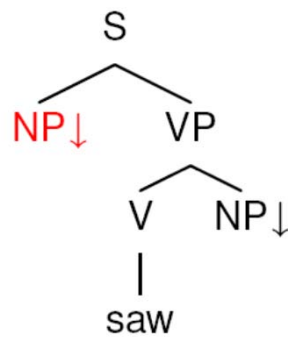
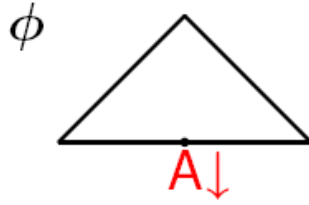
- Rewrite large (possibly lexicalized) subtrees in a single step

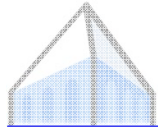


- Formally, a *tree-insertion grammar*
- Derivational ambiguity whether subtrees were generated atomically or compositionally
- Most probable *parse* is NP-complete



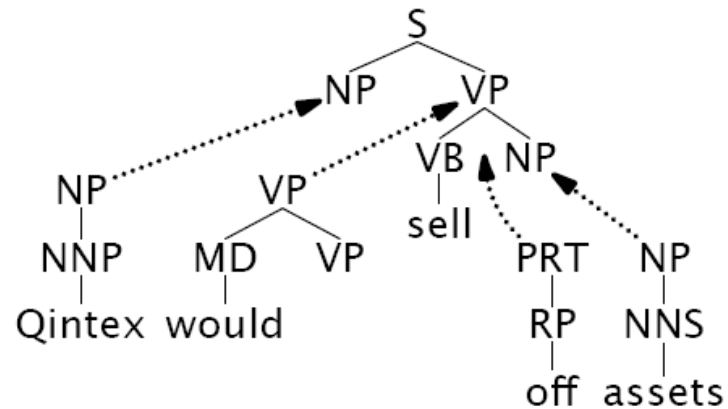
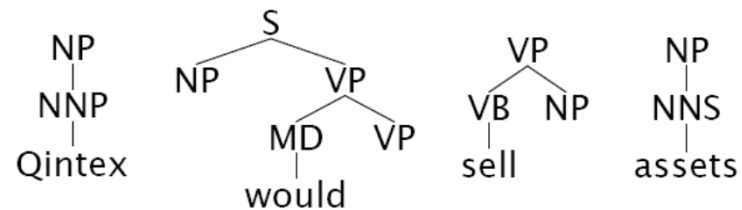
TIG: Insertion

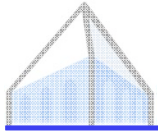




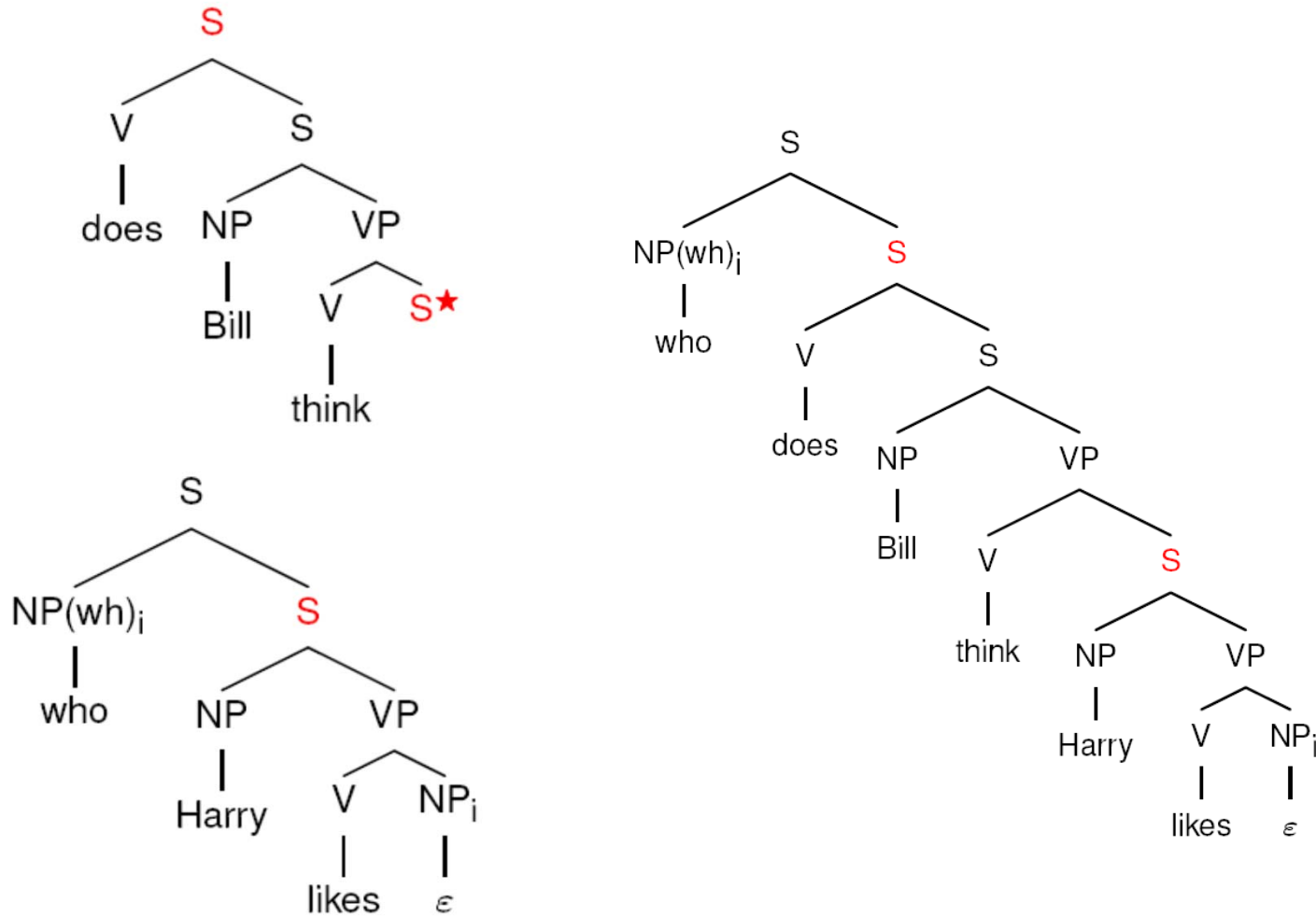
Tree-adjoining grammars

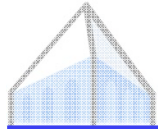
- Start with *local trees*
- Can insert structure with *adjunction* operators
- Mildly context-sensitive
- Models long-distance dependencies naturally
- ... as well as other weird stuff that CFGs don't capture well (e.g. cross-serial dependencies)





TAG: 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?)

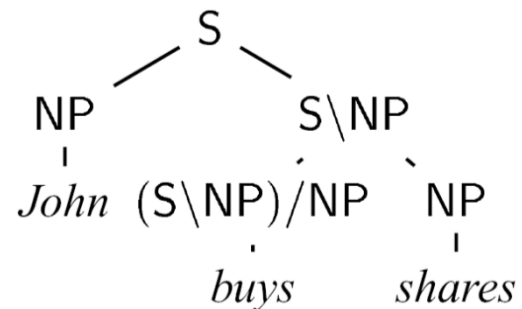
John \vdash NP

shares \vdash NP

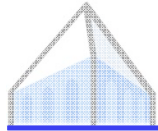
buys \vdash (S\NP)/NP

sleeps \vdash S\NP

well \vdash (S\NP)\(S\NP)

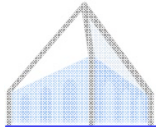


Empty Elements

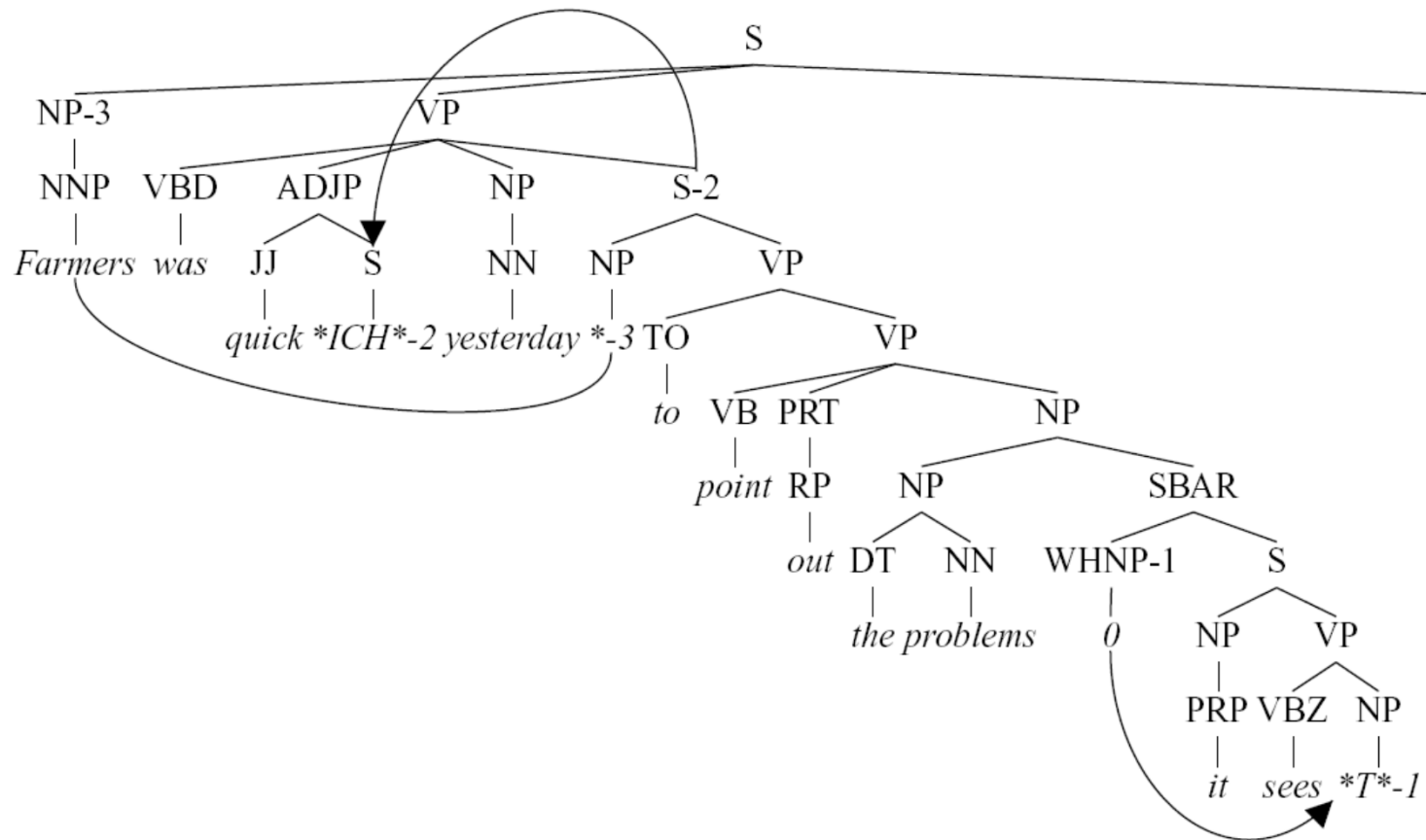


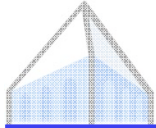
Empty Elements

- 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)

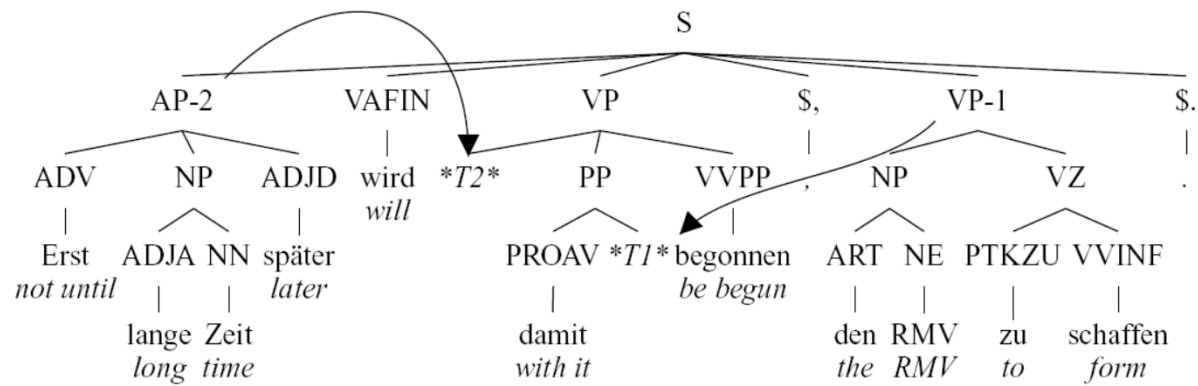


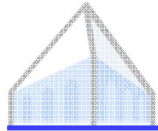
Example: English





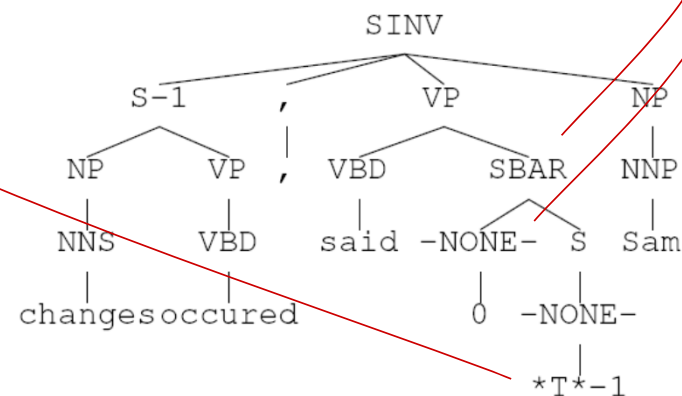
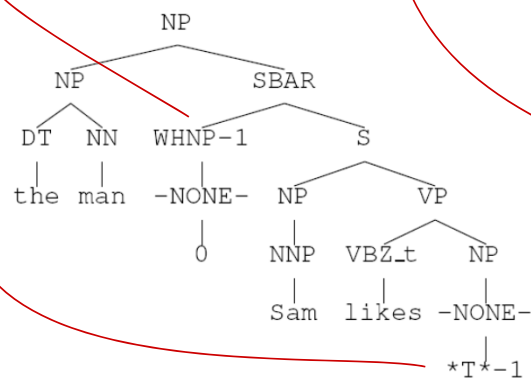
Example: German

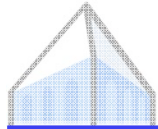




Types of Empties

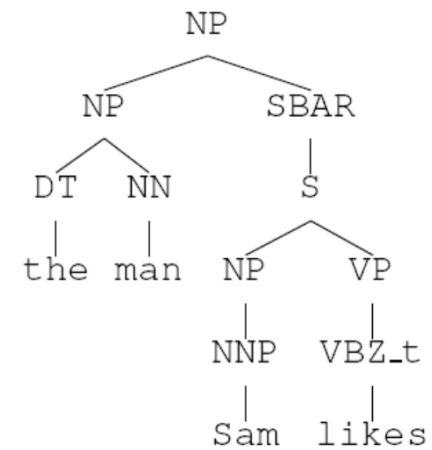
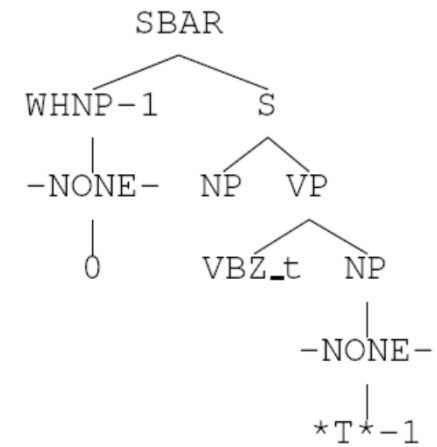
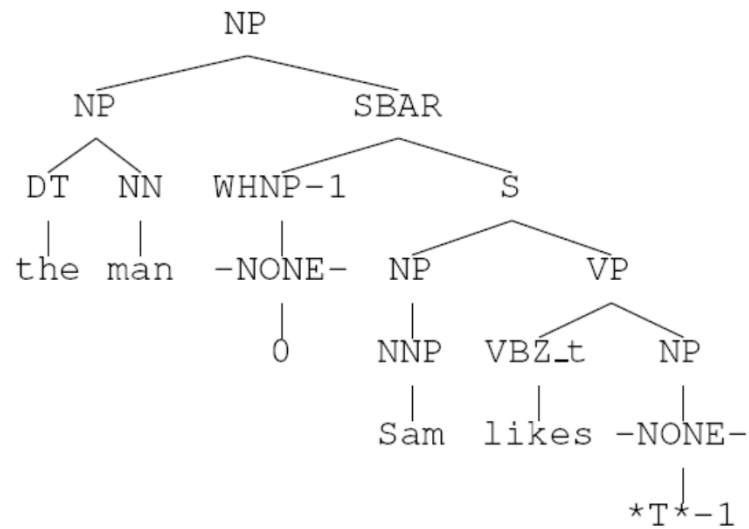
Antecedent	POS	Label	Count	Description
NP	NP	*	18,334	NP trace (e.g., <i>Sam was seen</i> *)
	NP	*	9,812	NP PRO (e.g., <i>* to sleep is nice</i>)
WHNP	NP	*T*	8,620	WH trace (e.g., <i>the woman who you saw</i> *T*)
		U	7,478	Empty units (e.g., <i>\$ 25</i> *U*)
		0	5,635	Empty complementizers (e.g., 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

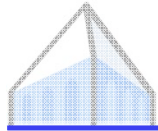




A Pattern-Matching Approach

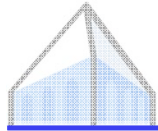
- [Johnson 02]





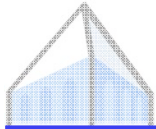
Pattern-Matching Details

- 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

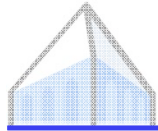
Count	Match	Pattern
5816	6223	(S (NP (-NONE- *)) VP)
5605	7895	(SBAR (-NONE- 0) S)
5312	5338	(SBAR WHNP-1 (S (NP (-NONE- *T*-1)) VP))
4434	5217	(NP QP (-NONE- *U*))
1682	1682	(NP \$ CD (-NONE- *U*))
1327	1593	(VP VBN_t (NP (-NONE- *)) PP)
700	700	(ADJP QP (-NONE- *U*))
662	1219	(SBAR (WHNP-1 (-NONE- 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	<i>P</i>	<i>R</i>	<i>f</i>	<i>P</i>	<i>R</i>	<i>f</i>
	(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

Semantic Roles



Semantic Role Labeling (SRL)

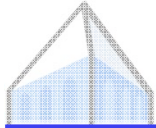
- Characterize clauses as *relations with roles*:

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

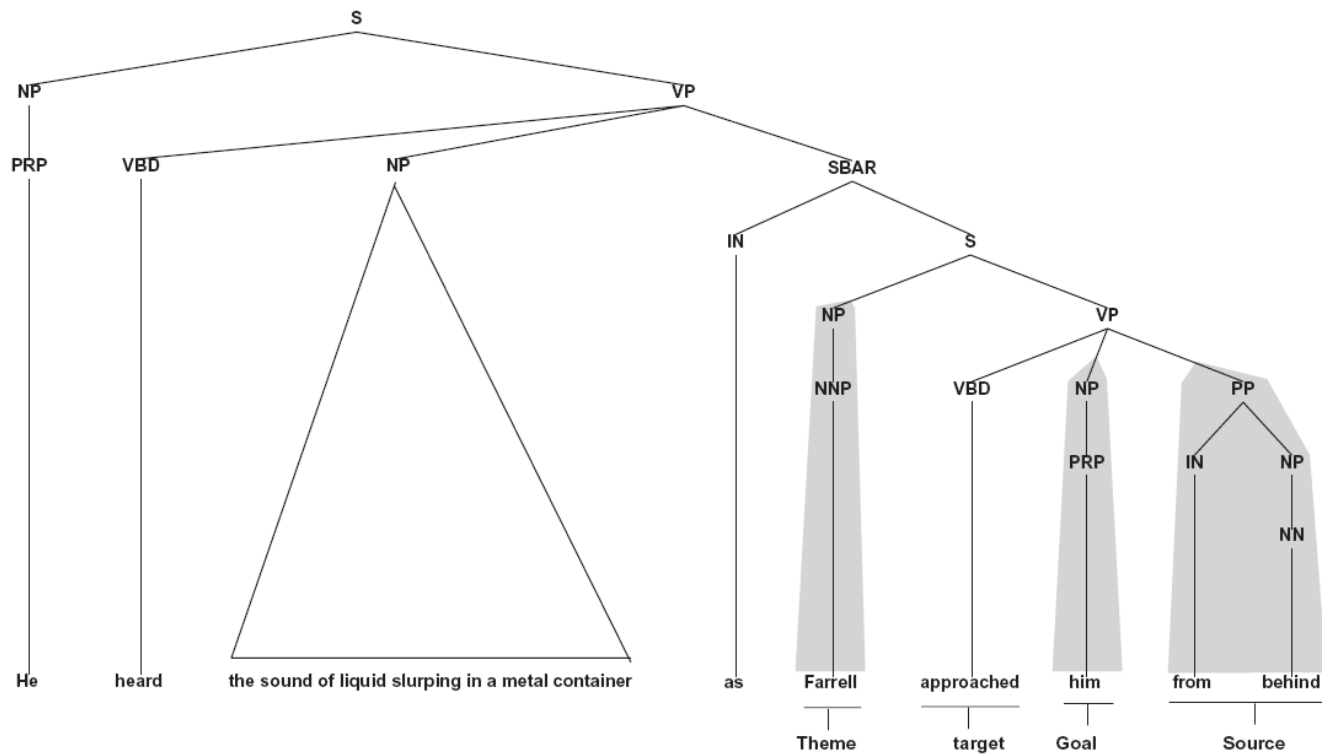
Holman would characterise this as **blaming** [*Evaluee* the poor] .

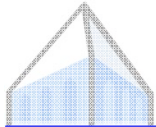
The letter quotes Black as saying that [*Judge* white and Navajo ranchers] misrepresent their livestock losses and **blame** [*Reason* everything] [*Evaluee* on coyotes] .

- Says 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

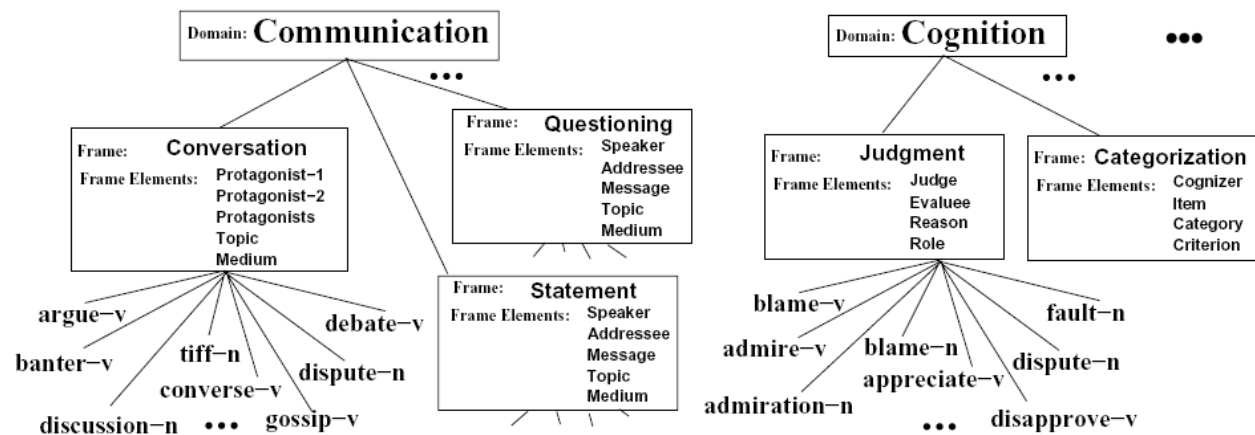


SRL Example

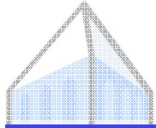




PropBank / FrameNet



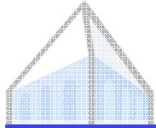
- FrameNet: roles shared between verbs
- PropBank: each verb has its own roles
- PropBank more used, because it's layered over the treebank (and so has greater coverage, plus parses)
- Note: some linguistic theories postulate fewer roles than FrameNet (e.g. 5-20 total: agent, patient, instrument, etc.)



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

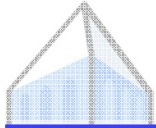


PropBank Example

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

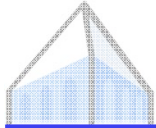
The Central Council of Church Bell Ringers aims *trace* to
improve relations with vicars. (wsj_0089)

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

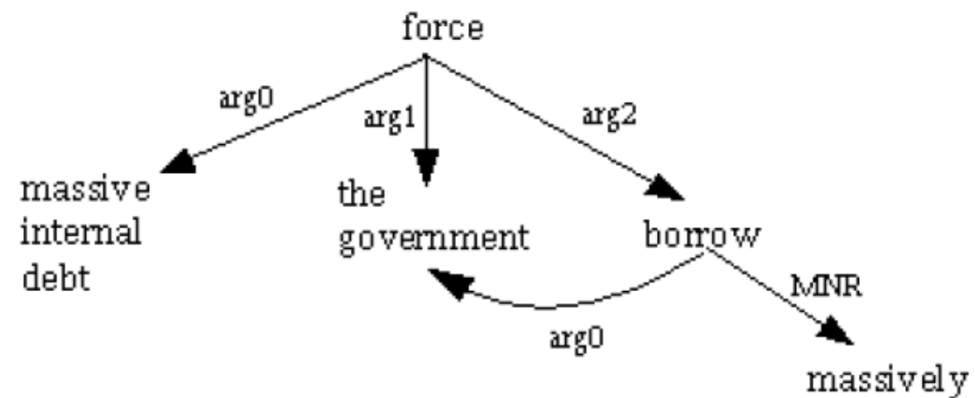
Banks have been aiming packages at the elderly.

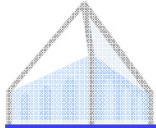
arg0: Banks
rel: aiming
arg1: packages
arg2: at the elderly



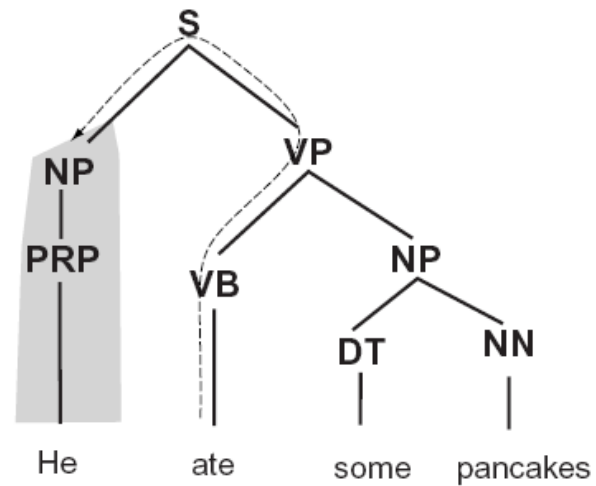
Shared Arguments

(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))...

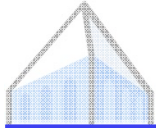




Path Features



<i>Path</i>	<i>Description</i>
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

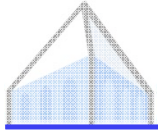


Results

- 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
 - Harder on automatic parses

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

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



Empties and SRL

