A simple pattern-matching algorithm for recovering empty nodes

Mark Johnson

Brown University

ACL'02, Philadelphia

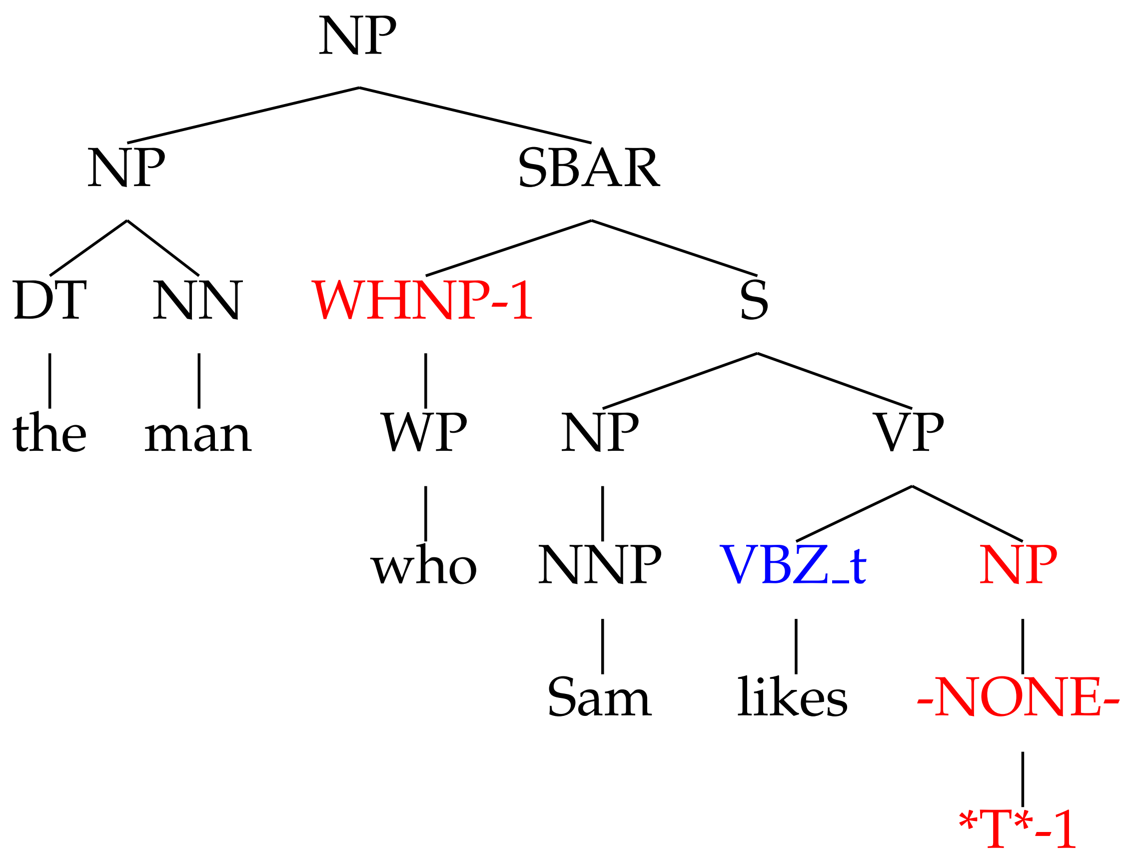
Thanks to Eugene Charniak and fellow BLLIPers

NSF grants DMS 0074276 and ITR IIS 0085940

Talk outline

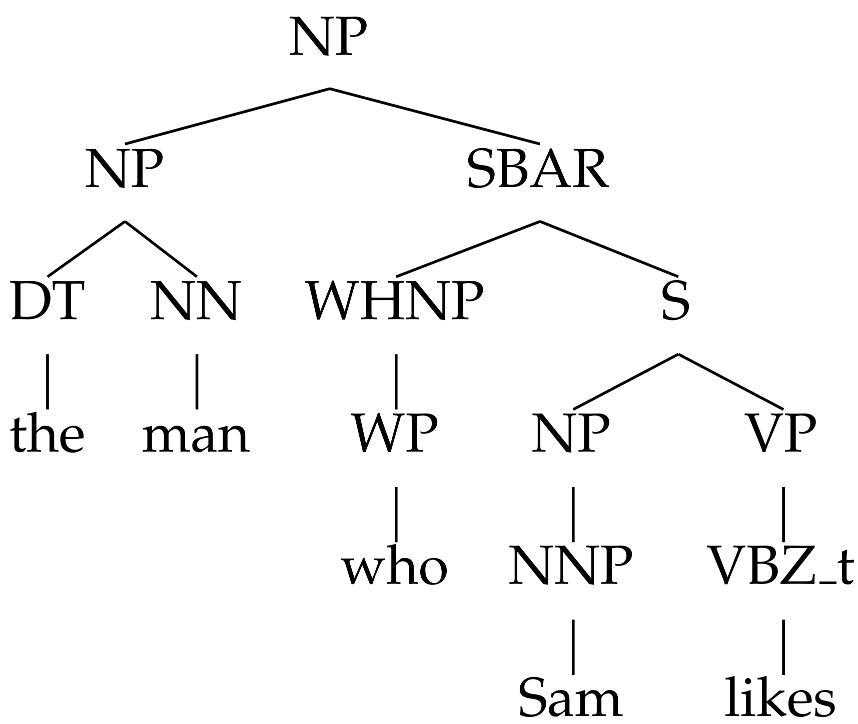
- Empty nodes in the Penn treebank representations
- A pattern-matching algorithm
- Evaluating empty node accuracy
- Evaluation on gold standard and parser trees

Empty nodes in Penn treebank



- *Empty nodes* and *co-indexation* indicate *non-local dependencies* that are important for *semantic interpretation*
- Likely to be important for *question-answering* and *machine translation*

Output of a statistical parser



- The output of most modern statistical parsers only encode *local dependencies*
  - Collins (1997) discusses recovering WH dependencies
  - SUBGs typically encode non-local dependencies

Other previous work on empty nodes

Generative syntax: Non-local dependencies are a major theme

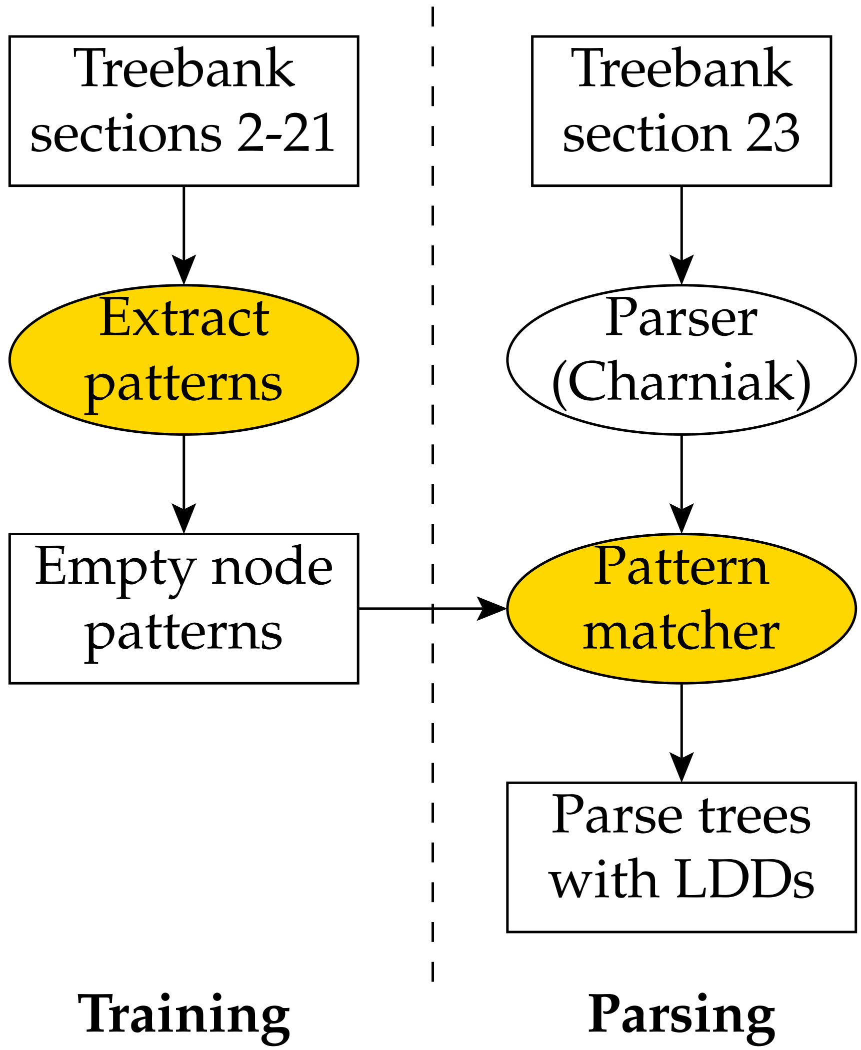
- Extremely complex theories
- Focuses on esoteric constructions
- Studies just a few kinds of non-local dependencies

Psycholinguistics: has studied interpretation of non-local dependencies

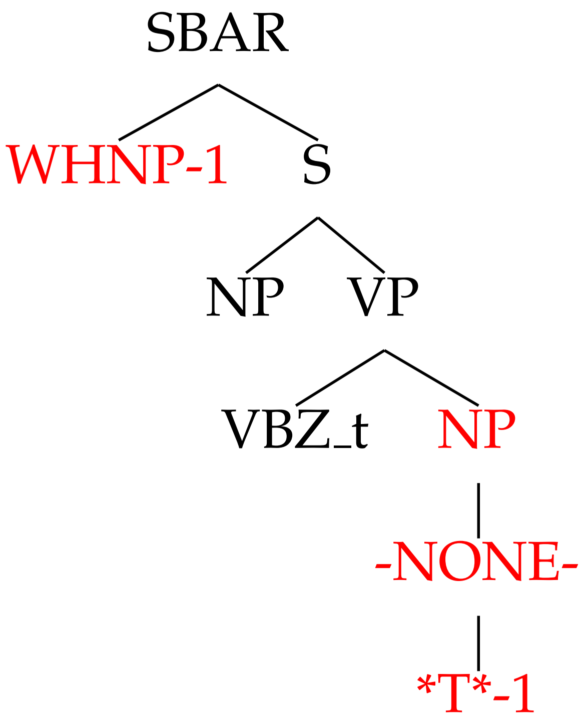
- Preferences for location of empty nodes
- How non-local dependencies affect complexity of sentence processing

- The *pattern-matching approach* described here is:
  - Theory neutral
  - Data-driven: trained from tree-bank⋆
  - Relatively straight-forward to implement
  - Can serve as a *base-line for more complex systems*

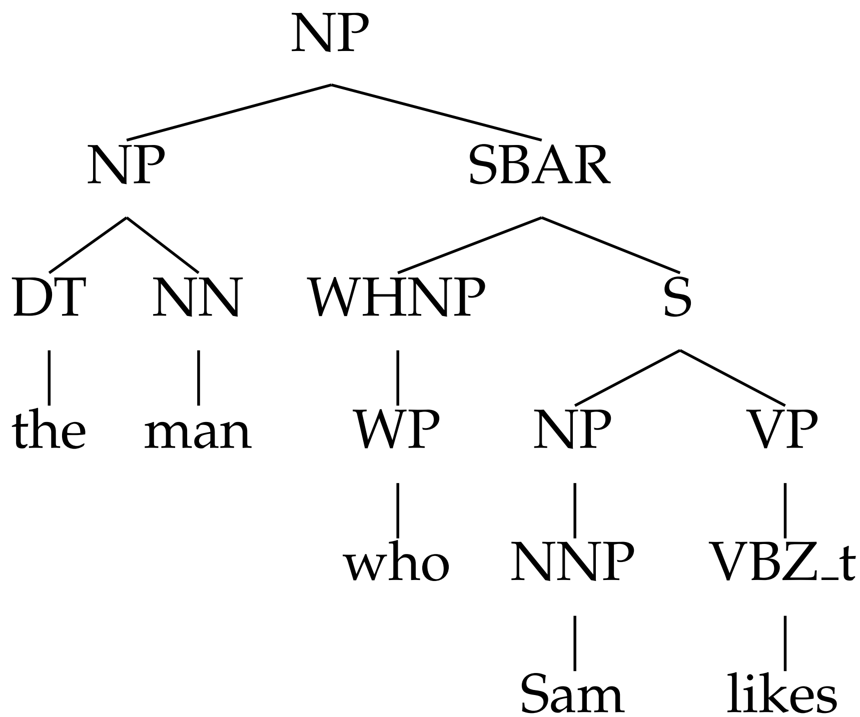
System architecture



Empty node insertion via pattern-matching



Pattern



Parser output

- Patterns extracted from Penn treebank training corpus (sections 2-21)
- Patterns matched against parser output
- A matching pattern suggests a long-distance dependency

Summary of empty nodes in Penn trees

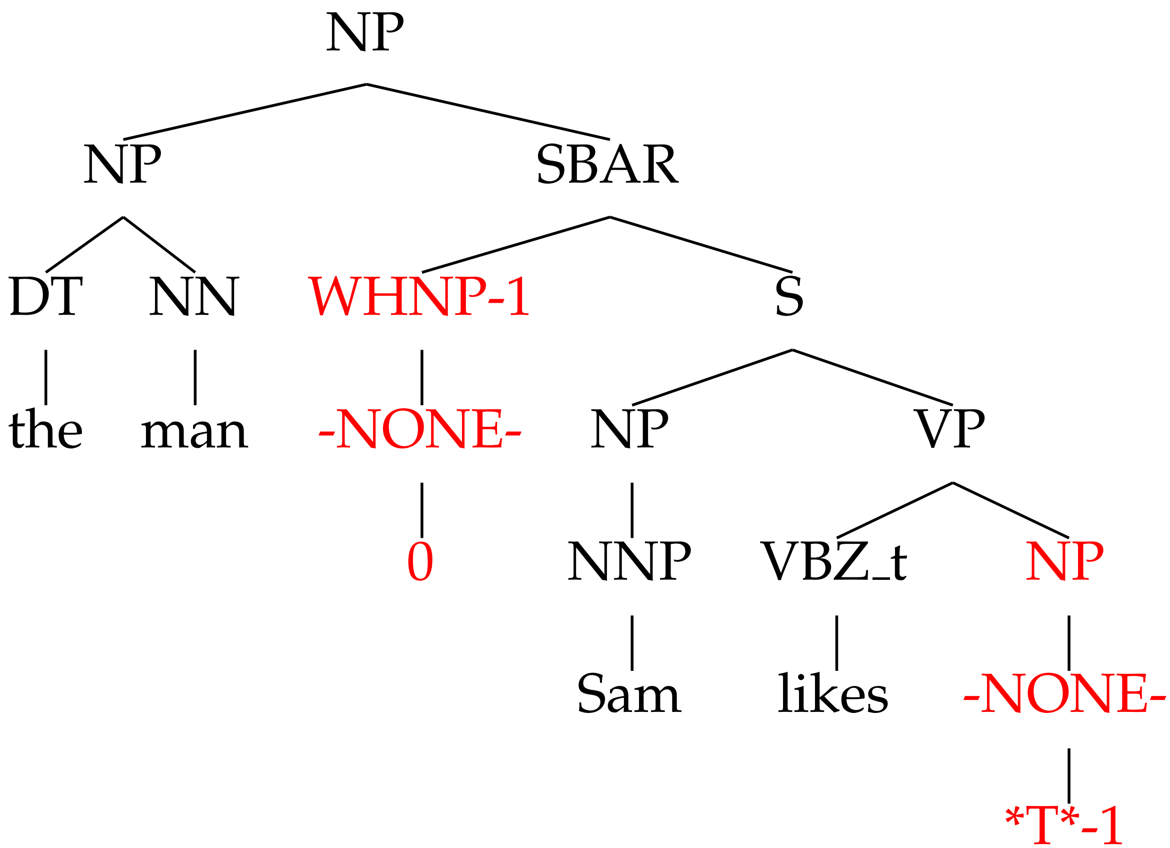
Antecedent	Category	Label	Count	Description
NP	NP	*	18,334	NP trace (Passive) *Sam was seen **
	NP	*	9,812	NP PRO (implied subject) ** to sleep is nice*
WHNP	NP	*T*	8,620	WH trace (questions, relative clauses) *the woman who you saw *T**
		U	7,478	Empty units *$ 25 *U**
		0	5,635	Empty complementizers *Sam said 0 Sasha snores*

Summary of empty nodes in Penn trees

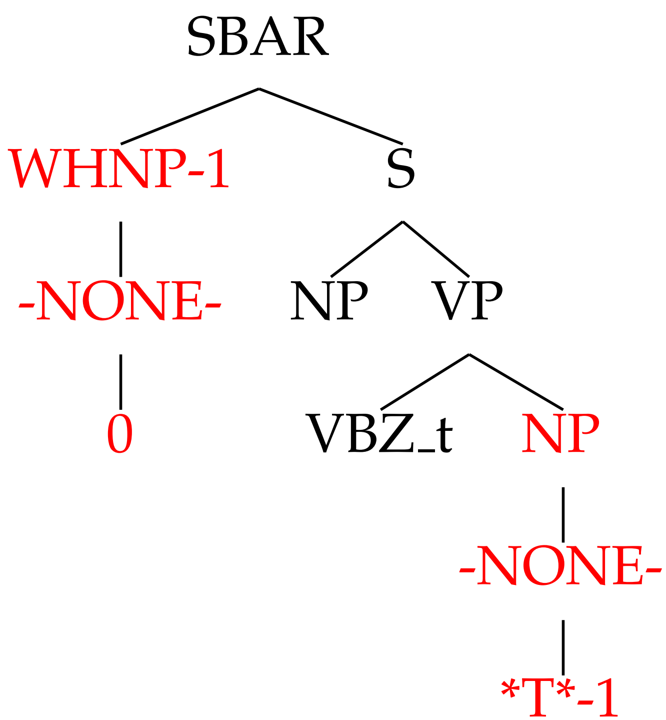
Antecedent	Category	Label	Count	Description
S	S	*T*	4,063	Moved clauses *Sam had to go, Sasha explained *T**
WHADVP	ADVP	*T*	2,492	WH-trace *Sam explained how to leave *T**
	SBAR		2,033	Empty clauses *Sam had to go, Sasha explained (SBAR)*
	WHNP	0	1,759	Empty relative pronouns *the woman 0 we saw*
	WHADVP	0	575	Empty relative pronouns *no reason 0 to leave*

- *Zipfian distribution of empty node types*

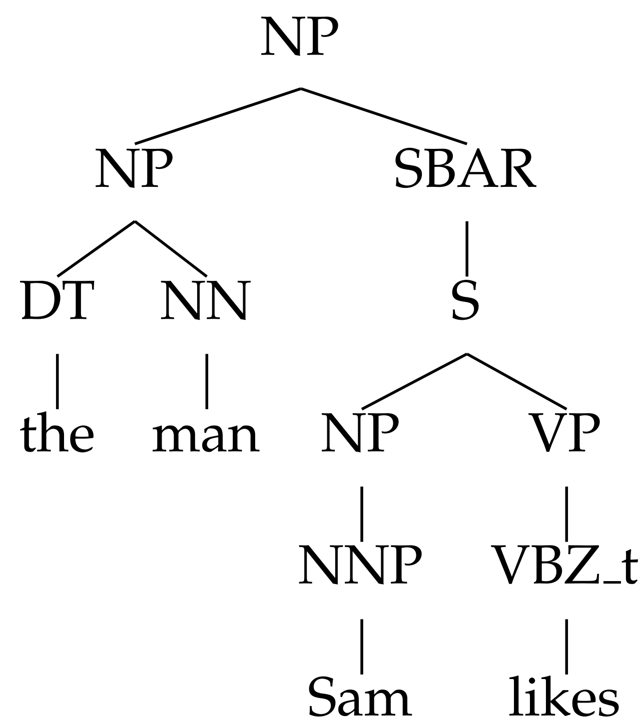
Two empty nodes in a long-distance dependency



Pattern and parser output

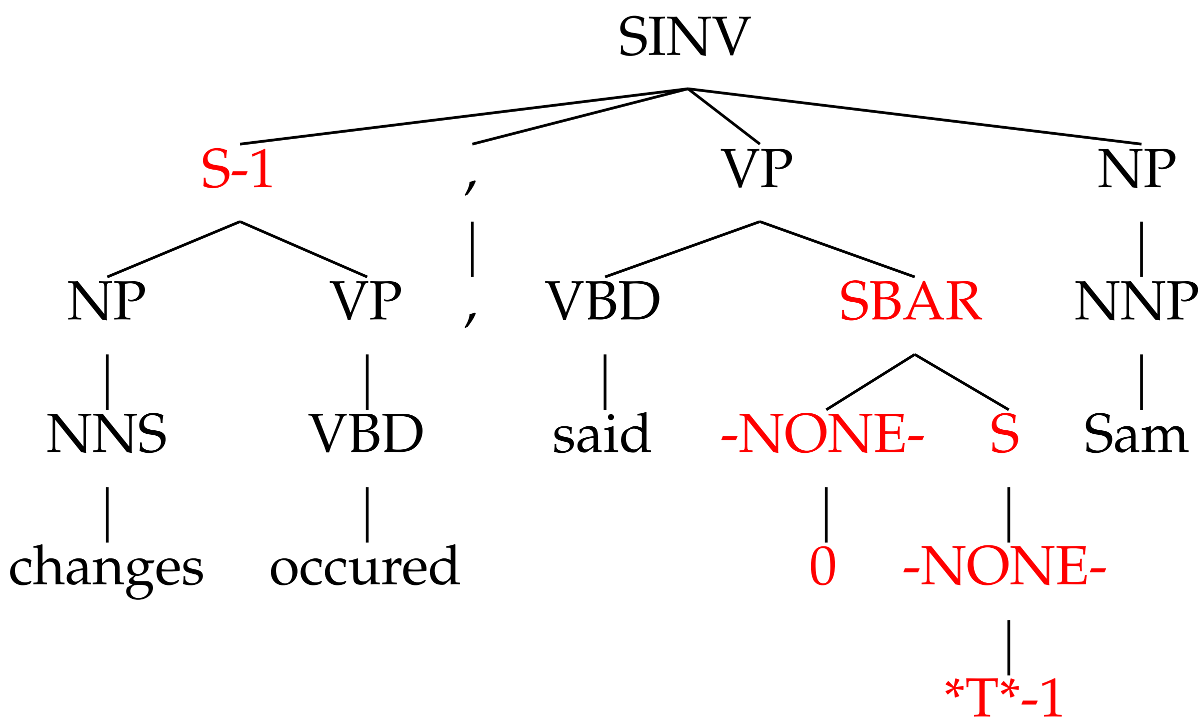


Pattern

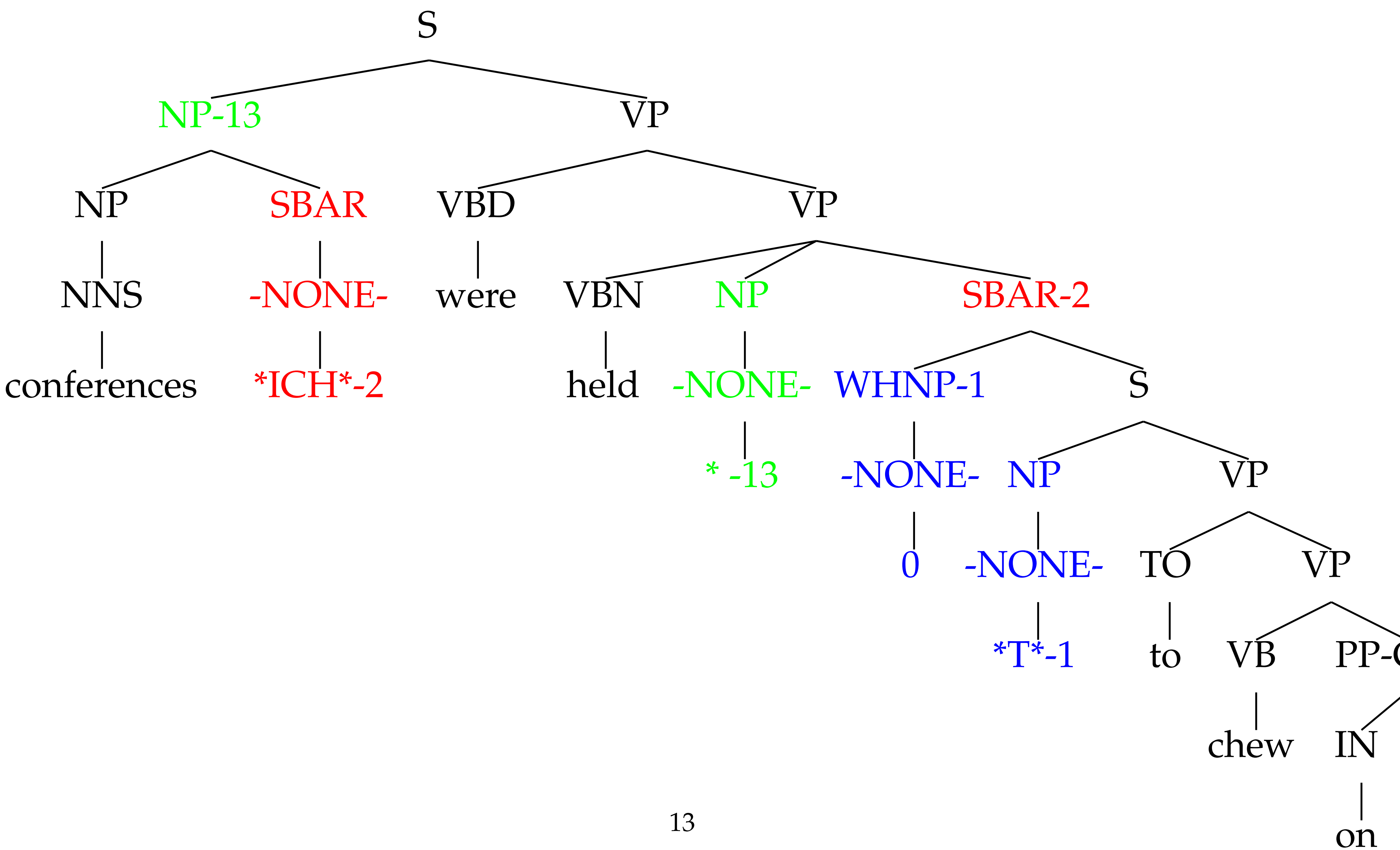


Parser output

Empty compound SBAR



Extraposition and adjunction



Tree preprocessing

Auxiliary POS replacement: The POS of auxiliary verbs *is*, *being*, etc. are replaced by AUX, AUXG, etc. (Charniak)

Transitivity relabelling: The POS labels of transitive verbs are suffixed "_t", e.g., *likes* is relabelled VBZ_t

- Transitivity is hypothesised to be a powerful cue to empty node placement
- Experiments on heldout data indicate this improves accuracy
- A verb is deemed transitive if it is followed by an NP with no function tag at least 50% of the time in the training corpus
- Morphological analysis may improve transitivity identification

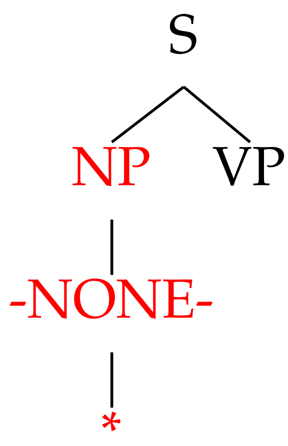
Patterns and matchings

- A pattern is *the minimal set of local trees* that connects each empty node with the nodes coindexed with it
- Indices are systematically renumbered*
- The implementation deals with *adjunction* and overlapping long-distance dependencies
  - Probably has a neglible effect on performance

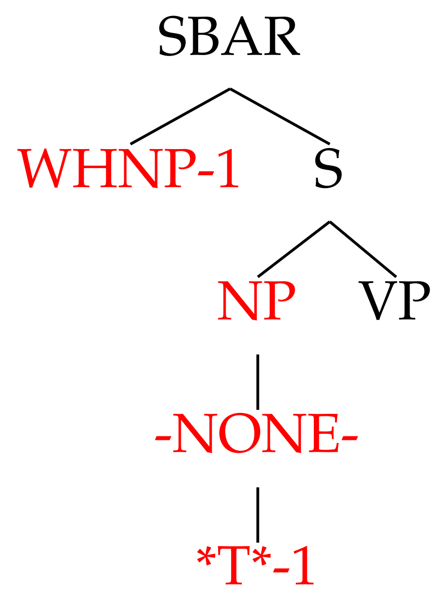
Empty node insertion

- Patterns are matched at each node in the tree
- Approximately 11,000 patterns
  - Pattern matching is speeded by indexing patterns on their topmost local tree
- Nodes in the tree to be matched are visited by a *preorder traversal*
  - Matching and insertion of deep pattern may destroy the context of a shallow one
  - Biases the algorithm in favor of deeper patterns

Overlapping patterns



The most common pattern



The third most common pattern

- The most common pattern will match every context that the third most common pattern matches (but not vice-versa)
- Preorder node traversal ensures that the third most common pattern gets a chance to match

Pattern extraction and selection

- Every pattern in training corpus is extracted
- For each pattern:
  - c: the number of times extracted
  - m: the number of times it matches some context in training corpus
    * Difficult to estimate because a larger pattern might destroy the context for a smaller one
  - If *discounted success probability* $< 1/2$ the pattern is discarded
    * Around 9,000 patterns remain after filtering
  - Patterns are *sorted* by depth (deep patterns first)
    * Exactly how patterns are sorted (e.g., frequency, discounted success probability) doesn't seem to matter

The most common patterns

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 .)

Empty node recovery evaluation

- Two different evaluation methods
  - *Standard Parseval evaluation:* evaluates empty node location, but not coindexation
  - *Extended evaluation:* evaluates both empty node location and coindexation
- Evaluate on *test trees without empty nodes* and on *parser output*

Standard Parseval evaluation: Nodes identified by a triple $\langle cat, left, right\rangle$ (note *left* = *right* for empty nodes)

- $G =$ set of empty nodes identified in gold-standard trees
- $T =$ set of trees produced by parser*

$$P = \frac{|G \cap T|}{|T|} \qquad R = \frac{|G \cap T|}{|G|} \qquad f = \frac{2\,P\,R}{P + R}$$

Empty node identification results

Empty node Category	Label	Section 23 P	R	f	Parser output 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

Evaluation of empty nodes and their antecedents

- Each empty node is identified by a *set of triples* $\langle cat, left, right\rangle$ corresponding to
  - the empty node itself
  - each node co-indexed with the empty node
- In order to "get the empty node right", the category and location of each of its antecedents must be recovered
  - Most empty nodes have zero or one antecedents
  - Stringent requirement, which also evaluates parser accuracy
  - Other measures (e.g., which only require identification of the head of the antecedent) yield very similiar results

Empty node and antecedent identification results

Empty node			Section 23			Parser output		
Antecedant	**POS**	**Label**	P	R	f	P	R	f
(Overall)			0.80	0.70	0.75	0.73	0.63	0.68
NP	NP	*	0.86	0.50	0.63	0.81	0.48	0.60
WHNP	NP	*T*	0.93	0.88	0.90	0.85	0.77	0.80
	NP	*	0.45	0.77	0.57	0.40	0.67	0.50
		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	S	*T*	0.98	0.83	0.90	0.96	0.79	0.87
WHADVP	ADVP	*T*	0.91	0.52	0.66	0.82	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

Discussion

- *Empty node identification* can be performed with reasonable accuracy
  - Performance drop-off on parser trees
  - Precision $\gg$ recall $\Rightarrow$ patterns may be too specialized
    - *Skeletal patterns* trade precision for recall, but leave f-score unchanged
- *Antecedent recovery* is considerably harder
  - Only half of the bound NP PRO are recovered!
    - Requires semantic/pragmatic information about interpretation
    - 10 pages of rules/examples about NP PRO indexing in tagging guidelines!
    - *Lexicalized patterns* ought to help, but didn't
    - More sophisticated classifiers (boosted decision stubs) had very similar performance to simple pattern matcher
  - Many long distance dependencies (e.g., WH-dependencies) can on average be reliably identified

Conclusions and Future Work

- This paper proposed two Parseval-style measures to evaluate empty node identification and antecedent identification
  - Restricted to Penn treebank style representation of long distance dependencies
- A simple pattern-matching post-processing approach to long-distance dependency identification works reasonably well
- Provides a baseline against which to evaluate more sophisticated systems
- Performance drop-off when using parser trees

  $\Rightarrow$ a single system that integrates parsing and long distance dependency identification may perform better