

Three Generative, Lexicalised Models for Statistical Parsing

Michael Collins

Proceedings of the 35th Annual meeting of the ACL

Revised and updated as

Head-Driven Statistical Models for Natural Language Parsing

Computational Linguistics 29(4) December 2003

Presented by Ken Hoetmer

Outline

- What does “generative” mean?
- Probabilistic context-free grammars
- Three parsing models
- Implementation issues
- Results
- Conclusions

The term “generative”

- Syntactic trees are “generated” by the top-down application of grammar rules.
- Parse trees can be represented by a series of decisions (so-called “history models”).
- Does NOT directly refer to:
 - ▶ generative grammar,
 - ▶ transformational grammar,
 - ▶ \bar{X} -bar theory,
 - ▶ government-binding theory,
 - ▶ the minimalist program,
 - ▶ or anything else Chomskyian...

Context-Free grammars

A context-free grammar (CFG) is a tuple $\langle N, T, S, R \rangle$ where:

- N is a set of non-terminal symbols,
- T is a set of terminal symbols,
- S is a start symbol,
- R is a set of rewrite rules $X \rightarrow \beta$ where $X \in N$ and $\beta \in (N \cup \Sigma)^*$.

Probabilistic CFGs

Every rule $X \rightarrow \beta$ has an associated probability $P(\beta | X)$.

Probability of a derivation :

$$P(T, S) = \prod_{i=1 \dots n} P(RHS_i | LHS_i)$$

Estimate rule probabilities using MLE:

$$P(\beta | X) = \frac{Count(X \rightarrow \beta)}{Count(X)}$$

Finding maximal trees

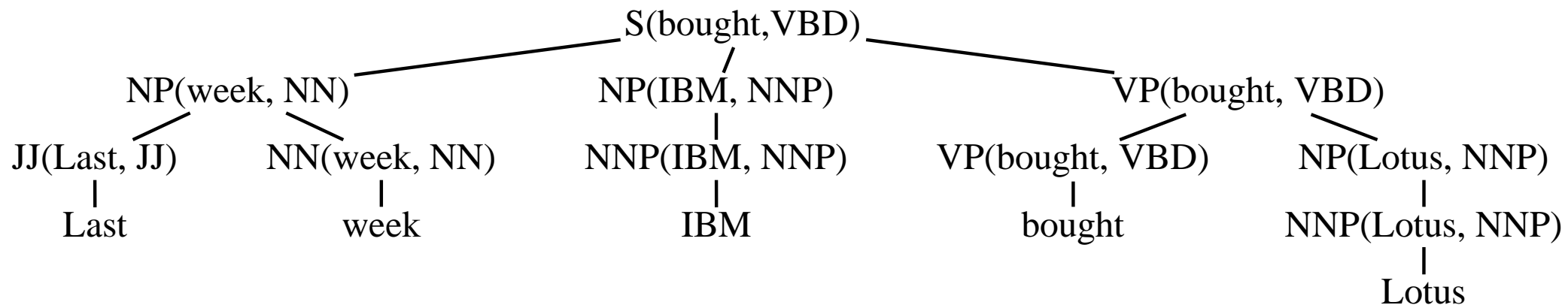
To find the most probable tree for a sentence S :

$$T_{best} = \arg \max_T P(T | S) = \arg \max_T \frac{P(T, S)}{P(S)} = \arg \max_T P(T, S)$$

T_{best} can be found using a trellis-based dynamic programming algorithm .

Lexicalised context-free grammars

Capture word-specific properties by associating lexical information with each non-terminal.



Lexicalised PCFG rule probabilities

Naive MLE:

$$P(\text{NP}(\text{week}, \text{NN}) \text{ NP}(\text{IBM}, \text{NNP}) \text{ VP}(\text{bought}, \text{VBD}) \mid \text{S}(\text{bought}, \text{VBD})) = \frac{\text{Count}(\text{S}(\text{bought}, \text{VBD}) \rightarrow \text{NP}(\text{week}, \text{NN})\text{NP}(\text{IBM}, \text{NNP})\text{VP}(\text{bought}, \text{VBD}))}{\text{Count}(\text{S}(\text{bought}, \text{VBD}))}$$

The problem?

$$\begin{aligned} |\text{Lexicalised Non-terminals}| &= \\ |\text{Vocabulary}| \times |\text{POS Tags}| \times |\text{Non-terminals}| \end{aligned}$$

Collins's 3 parsing models

Three generative, lexicalised models proposed by Collins:

- Model 1: Generative model with lexicalised heads
- Model 2: Model 1 plus complement/adjunct distinction and subcategorisation
- Model 3: Model 2 plus traces and wh-movement

Model 1: Introduction

Associate a word w and POS t with each non-terminal. Collins writes each non-terminal as $X(x)$ where $x = \langle w, t \rangle$ and X is the traditional non-terminal constituent label.

Each rule becomes:

$$P(h) \rightarrow L_n(l_n) \dots L_1(l_1)H(h)R_1(r_1) \dots R_m(r_m)$$

where H is the head of the phrase and $L_1 \dots L_n$ and $R_1 \dots R_m$ are the left and right modifiers of H .

Estimating $P(RHS|LHS)$ directly is impossible due to the number of lexical heads.

Therefore, Collins reduces the search space by assuming each modifier is dependant only on the head of the rule.

Model 1: Rule generation

Generating the rule becomes a three step process:

- Generate the head with probability $P_H(H | P, h)$
- Generate right hand modifiers with probability $\prod_{i=1 \dots m+1} P_R(R_i(r_i) | P, h, H)$ where $R_{m+1}(r_{m+1}) = \text{STOP}$.
- Generate left hand modifiers with probability $\prod_{i=1 \dots n+1} P_L(L_i(l_i) | P, h, H)$ where $L_{n+1}(l_{n+1}) = \text{STOP}$.

Model 1: Rule generation

$S(\text{brought}) \rightarrow NP(\text{week}) NP(\text{Marks}) VP(\text{bought})$

is generated as:

$P_h(VP|S, \text{bought}) \times$

$P_l(NP(\text{Marks})|S, VP, \text{bought}) \times P_l(NP(\text{week})|S, VP, \text{bought}) \times$

$P_l(STOP|S, VP, \text{bought}) \times$

$P_r(STOP|S, VP, \text{bought})$

Model 1: Rule generation

Notice: Collins has made a 0^{th} -order Markov assumption:

$$P_l(L_i(l_i)|H, P, h, L_1(l_1) \dots L_{i-1}(l_{i-1})) = P_l(L_i(l_i)|H, P, h)$$

$$P_r(R_i(r_i)|H, P, h, R_1(r_1) \dots R_{i-1}(r_{i-1})) = P_r(R_i(r_i)|H, P, h)$$

However, assuming top-down generation, $P_l(L_i(l_i))$ and $P_r(R_i(r_i))$ could be conditioned on any set of previously generated modifiers OR (properties of) the subtrees they dominate.

Model 1: Adding a distance measure

Collins introduces surface string distance measures Δ_l and Δ_r .

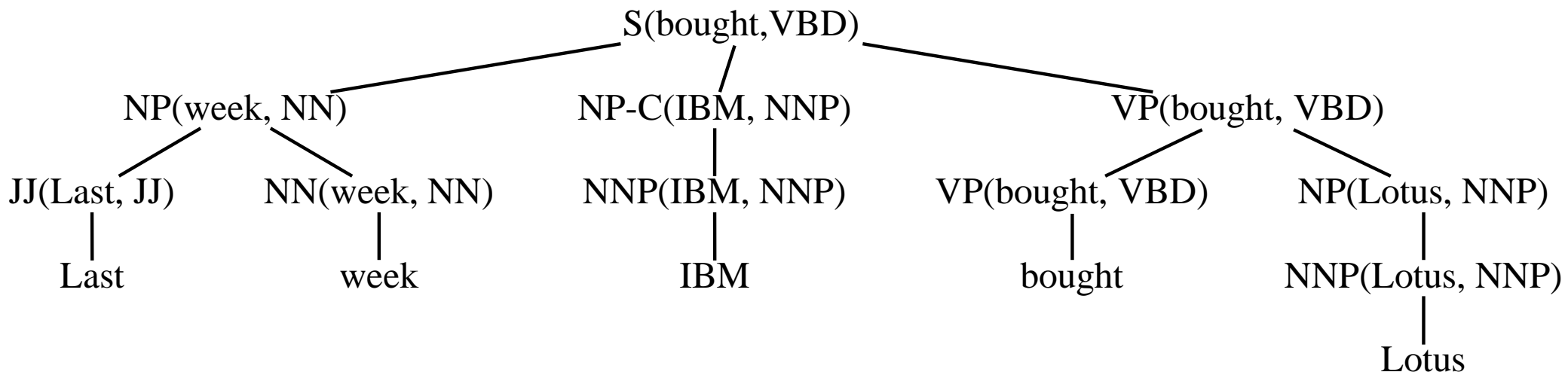
Are not REAL distance - functions are heuristics based on direction, adjacency, verb positions, and punctuation.

The model becomes:

$$P_l(L_i(l_i)|H, P, h, L_1(l_1) \dots L_{i-1}(l_{i-1})) = P_l(L_i(l_i)|H, P, h, \Delta_l(i-1))$$

$$P_r(R_i(r_i)|H, P, h, R_1(r_1) \dots R_{i-1}(r_{i-1})) = P_r(R_i(r_i)|H, P, h, \Delta_r(i-1))$$

Model 2: Classifying modifiers



Model 2: Identifying complements

Add a “-C” suffix to non-terminals in the training data for which:

- The non-terminal is an NP, SBAR, or S whose parent is an S, an NP, SBAR, S, or VP whose parent is a VP, or an S whose parent is an SBAR.
- The non-terminal must not have one of the semantic tags: ADV, VOC, BNF, DIR, EXT, LOC, MNR, TMP, CLR, or PRP.

Also mark the first child following a PP head as a complement.

Model 2: Subcategorization frames

Define a subcategorization frame as a bag of non-terminals.

Then:

- Generate the head with probability $P_H(H | P, h)$,
- Choose right and left subcat frames with probabilities $P_{lc}(LC | P, H, h)$ and $P_{rc}(RC | P, H, h)$,
- Generate right hand modifiers with probability $P_r(R_i(r_i) | H, P, h, \Delta_r(i - 1), RC)$,
- Generate left hand modifiers with probability $P_l(L_i(l_i) | H, P, h, \Delta_l(i - 1), LC)$.

Model 2: Example

$P(S(\text{bought}) \rightarrow NP(\text{week}) NP-C(\text{Marks}) VP(\text{bought}))$

becomes

$$P_h(VP | S, \text{bought}) \times$$

$$P_{lc}(NP-C | S, VP, \text{bought}) \times P_{rc}(\{\} | S, VP, \text{bought}) \times$$

$$P_l(NP-C(\text{Marks}) | S, VP, \text{bought}, \{NP-C\}) \times$$

$$P_l(NP(\text{week}) | S, VP, \text{bought}, \{\}) \times$$

$$P_l(STOP | S, VP, \text{bought}, \{\}) \times$$

$$P_r(STOP | S, VP, \text{bought}, \{\})$$

Model 3: Modeling traces and wh-Movement

Subject extraction The store (SBAR which TRACE bought Brooks Brothers)

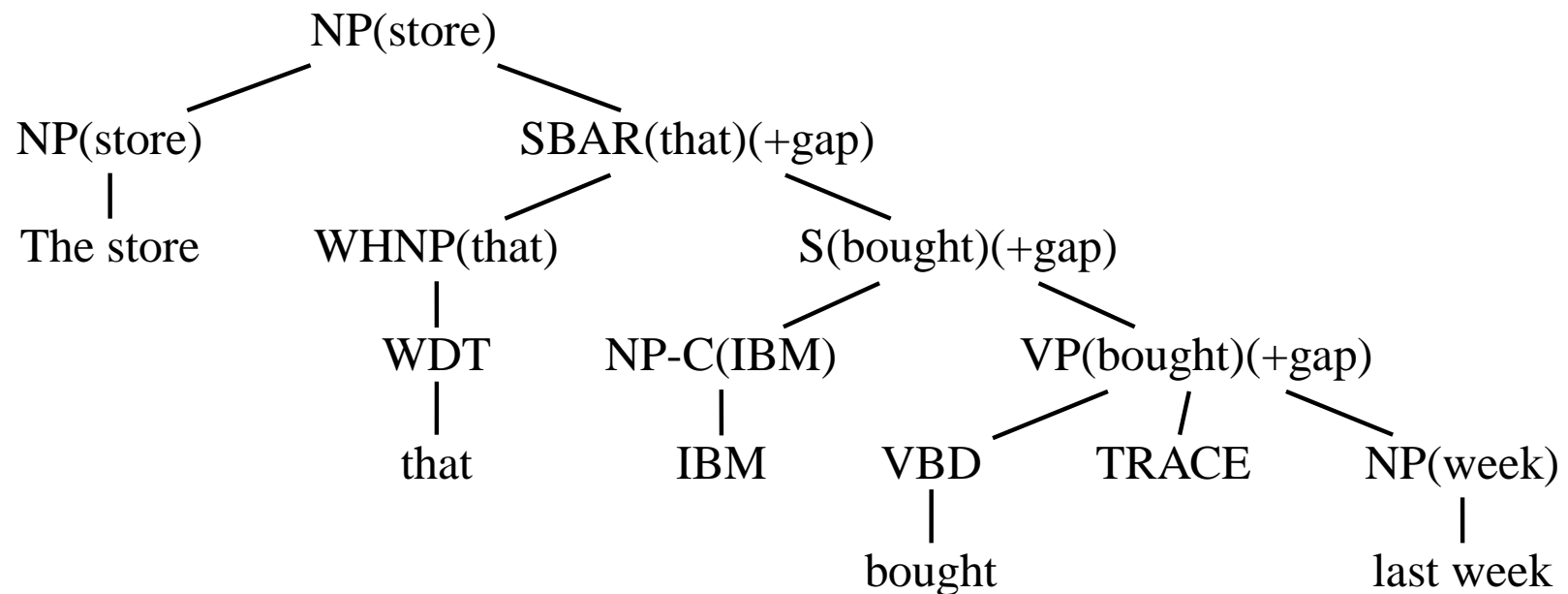
Object extraction The store (SBAR which Marks bought TRACE)

PP extraction The store (SBAR which Marks bought Brooks Brothers from TRACE)

Possible to write rules to identify traces? Yes... But Chomsky's attempt covered 92 pages!

Solution? A *gap* feature like those used in feature structure grammars.

Model 3: Gap features



Model 3: Passing gaps

3 ways to pass a gap to the RHS:

- Pass to the head of the phrase
- Pass to a left or right modifier
- Discharge as a trace argument to a left or right modifier

How? Introduce $P_g(G | P, h, H)$ where $G \in \{\text{Head, Left, Right}\}$

Model 3: Rule example

$P(\text{SBAR}(\text{that})(+\text{gap}) \rightarrow \text{WHNP}(\text{that}) \text{S-C}(\text{bought})(+\text{gap}))$

has probability:

$$P_h(\text{WHNP} | \text{SBAR}, \text{that}) \times$$

$$P_g(\text{Right} | \text{SBAR}, \text{WHNP}, \text{that}) \times$$

$$P_{lc}(\{\} | \text{SBAR}, \text{WHNP}, \text{that}) \times P_{rc}(\{\text{S-C}\} | \text{SBAR}, \text{WHNP}, \text{that}) \times$$

$$P_r(\text{S-C}(\text{bought})(+\text{gap}) | \text{SBAR}, \text{WHNP}, \text{that}, \{\text{S-C}, +\text{gap}\}) \times$$

$$P_r(\text{STOP} | \text{SBAR}, \text{WHNP}, \text{that}, \{\}) \times P_l(\text{STOP} | \text{SBAR}, \text{WHNP}, \text{that}, \{\})$$

Implementation: Parameter estimation

3 back-off levels:

- Level 1: all parameters included
- Level 2: remove conditioning on lexical item
- Level 3: remove conditioning on POS tag

P_{lc} example:

- Level 1: $e_1 = P_{lc}(LC | P, H, w, t)$
- Level 2: $e_2 = P_{lc}(LC | P, H, t)$
- Level 3: $e_3 = P_{lc}(LC | P, H)$

Implementation: Parameter estimation

Interpolate probabilities using Witten-Bell smoothing

$$e = \lambda_1 e_1 + (1 - \lambda_1)(\lambda_2 e_2 + (1 - \lambda_2)e_3)$$

Say $e_1 = \frac{n_1}{f_1}$ and let u_1 be number of *distinct* occurrences of f_1 .

$$\text{Then, } \lambda_1 = \frac{f_1}{f_1 + 5u_1} \quad \lambda_2 = \frac{f_2}{f_2 + 5u_2}$$

The value 5 is determined “empirically”.

Implementation

Words with training frequency < 6 replaced with POS tag “UNKNOWN”.

POS Tagging is NOT a preprocessing step: The models statistically select POS tags.

Parser implementation:

- Standard PCFG dynamic programming algorithm.
- Employs a beam search strategy: discard chart entries with $P(\dots) < \alpha$.
- Complexity $O(n^5)$ where $n = \text{length of sentence}$.

Results: Training and evaluation

Training Penn Treebank WSJ sections 02-21 (about 40,000 sentences)

Testing Penn Treebank WSJ section 23 (2416 sentences)

Evaluation PARSEVAL metrics(labeled precision, labeled recall, crossing brackets)

$$\text{Labeled precision} = \frac{\textit{number of correct constituents in proposed parse}}{\textit{number of constituents in proposed parse}}$$

$$\text{Labeled recall} = \frac{\textit{number of correct constituents in proposed parse}}{\textit{number of constituents in tree parse}}$$

Crossing Brackets = number of constituents which violate constituent boundaries with a constituent in the treebank parse.

Results: 3 Models

Sentence length ≤ 40 Words (2245 sentences)

Model	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 95)	84.6%	84.9%	1.26	56.6%	81.4%
(Collins 96)	85.8%	86.3%	1.14	59.9%	83.6%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%
Model 2	88.1%	88.6%	0.91	66.5%	86.9%
Model 3	88.1%	88.6%	0.91	66.4%	86.9%

Results: 3 Models

Sentence length ≤ 100 Words (2416 sentences)

Model	LR	LP	CBs	0 CBs	≤ 2 CBs
(Magerman 95)	84.0%	84.3%	1.46	54.0%	78.8%
(Collins 96)	85.3%	85.7%	1.32	57.2%	80.8%
Model 1	86.8%	87.6%	1.11	63.1%	84.1%
Model 2	87.5%	88.1%	1.07	63.9%	84.6%
Model 3	87.5%	88.1%	1.07	63.9%	84.6%

Previous work

- Collins 96: Similar to Model 1 but generates lexical item / head pair.
- Charniak 95: Similar rule breakdown but generates entire rule instead of the head.
- Eisner 96: 3 different dependency models.
- Magerman 95, Jelinek et al. 94: History based approach using decision trees.

Conclusions

- Statistical parsing is viable for Penn treebank grammars.
- Collins's models beat other state of the art parsers.
- Nice example of probabilistic treatment of subcategorisation and traces.
- What are the effects of the smoothing algorithm and beam size?

Previous work

Collins 96

Charniak 95

Eisner 96

Magerman 95, Jelinek et al. 94

Collins 2000: Model refinements

- Nonrecursive NPs
- Coordination
- Punctuation
- Empty (PRO) subjects
- Punctuation constraints

Results: Model Refinements

≤ 40 Words (2245 sentences)

Model	LR	LP	CBs	0 CBs	≤ 2 CBs
Magerman 1995	84.6%	84.9%	1.26	56.6%	81.4%
Collins 1996	85.8%	86.3%	1.14	59.9%	83.6%
Charniak 1997	87.5%	87.4%	1.00	62.1%	86.1%
Model 1	87.4%	88.1%	0.96	65.7%	86.3%
Model 2	88.1%	88.6%	0.91	66.5%	86.9%
Model 3	88.1%	88.6%	0.91	66.4%	86.9%
Charniak 2000	90.1%	90.1%	0.74	70.1%	89.6%
Collins 2000	90.1%	90.4%	0.74	70.7%	89.6%

Results: Model refinements

≤ 100 Words (2416 sentences)

Model	LR	LP	CBs	0 CBs	≤ 2 CBs
Magerman 1995	84.0%	84.3%	1.46	54.0%	78.8%
Collins 1996	85.3%	85.7%	1.32	57.2%	80.8%
Charniak 1997	86.7%	86.6%	1.20	59.5%	83.2%
Model 1	86.8%	87.6%	1.11	63.1%	84.1%
Model 2	87.5%	88.1%	1.07	63.9%	84.6%
Model 3	87.5%	88.1%	1.07	63.9%	84.6%
Charniak 2000	89.6%	89.5%	0.88	67.6%	87.7%
Collins 2000	89.6%	89.9%	0.87	68.3%	87.7%

A closer look at the parses returned
