{Probabilistic|Stochastic} Context-Free Grammars (PCFGs)

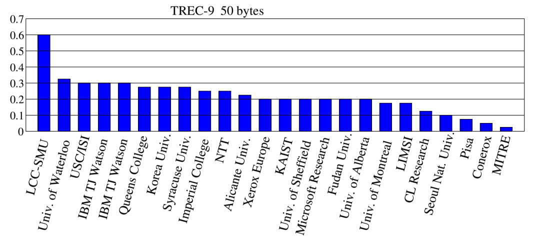
FSNLP, chapter 11

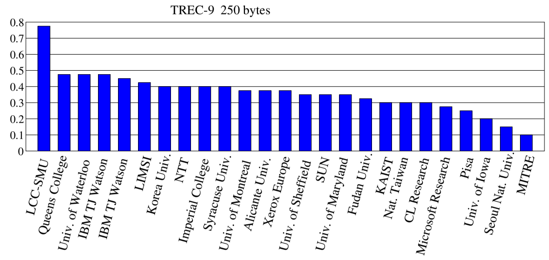
Christopher Manning and Hinrich Schütze

© 1999–2002

Question answering from text

- TREC 8+ QA competition (1999–; it's ongoing): an idea originating from the IR community
- With massive collections of on-line documents, manual translation of textual information into knowledge bases covering large numbers of domains is impractical: We want to answer questions from textbases
- Evaluated output is 5 answers of 50/250 byte snippets of text drawn from a 3 Gb text collection, and required to contain at least one concept of the semantic category of the expected answer type. (Until 2002. IR think: Suggests the use of named entity recognizers.)
- Get reciprocal points for highest correct answer.





Pasca and Harabagiu (2001) demonstrates the value of sophisticated NLP processing

- Good IR is needed: paragraph retrieval based on SMART
- Large taxonomy of question types and expected answer types is crucial
- Parsing: A statistical parser (modeled on Collins 1997) is used to parse questions, relevant text for answers, *and* WordNet to build a knowledge base for reasoning
- Controlled query expansion loops (morphological, lexical synonyms, and semantic relations) are all important in retrieving the correct answer.
- Answer ranking by ML method based on this information surpasses IR-style empirical methods.

Question Answering Example (1)

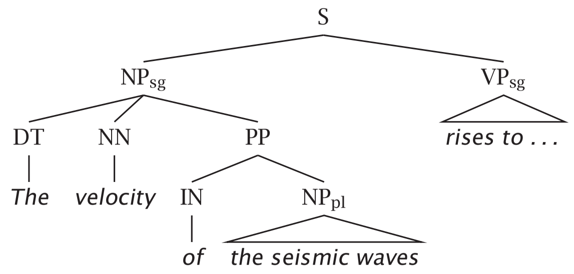
- Q261: What company sells most greetings cards ?
- sells(ORGANIZATION, cards(greeting), most)
- "Hallmark remains the largest maker of greeting cards"
  maker(ORGANIZATION(Hallmark), cards(greeting), largest)
- Need an entailment between *producing*, or *making* and *selling* goods
- Derived from WordNet, since synset *make, produce, create* has the genus **manufacture**, defined in the gloss of its nominalization as (*for*) *sale*
- Also, need *most* ≈ *largest*
- Therefore the semantic form of question Q261 and its illustrated answer are similar

Question Answering Example (2)

- How hot does the inside of an active volcano get ?
- get(TEMPERATURE, inside(volcano(active)))
- "lava fragments belched out of the mountain were as hot as 300 degrees Fahrenheit"
- fragments(lava, TEMPERATURE(degrees(300)),
  belched(out, mountain))
- □ volcano ISA mountain
  □ lava ISPARTOF volcano □ lava inside volcano
  □ fragments of lava HAVEPROPERTIESOF lava
- The needed semantic information is available in WordNet definitions, and was successfully translated into a form that can be used for rough 'proofs'

Why we need recursive phrase structure

- *The velocity of the seismic waves rises to . . .*
- Kupiec (1992): HMM tagger goes awry: *waves* → verb



PCFGs

A PCFG G consists of the usual parts of a CFG

- A set of terminals, $\{w^k\}, k = 1, \dots, V$
- A set of nonterminals, $\{N^i\}, i = 1, \dots, n$
- A designated start symbol, N^1
- A set of rules, $\{N^i \rightarrow \zeta^j\}$, (where ζ^j is a sequence of terminals and nonterminals)

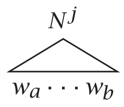
and

- A corresponding set of probabilities on rules such that:

$$\forall i \quad \sum_j P(N^i \rightarrow \zeta^j) = 1$$

PCFG notation

Sentence: sequence of words $w_1 \cdots w_m$
w_{ab}: the subsequence $w_a \cdots w_b$
N^i_{ab}: nonterminal N^i dominates $w_a \cdots w_b$



$N^i \stackrel{*}{\Longrightarrow} \zeta$: Repeated derivation from N^i gives ζ.

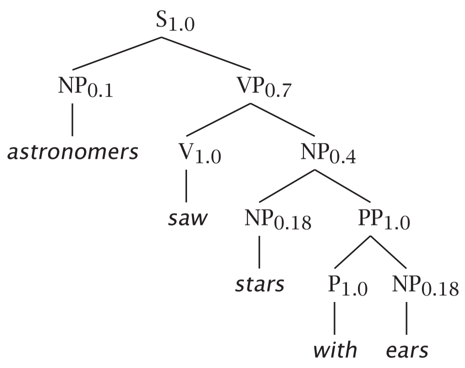
PCFG probability of a string

$$\begin{aligned} P(w_{1n}) &= \sum_t P(w_{1n}, t) \quad t \text{ a parse of } w_{1n} \\ &= \sum_{\{t:\text{yield}(t)=w_{1n}\}} P(t) \end{aligned}$$

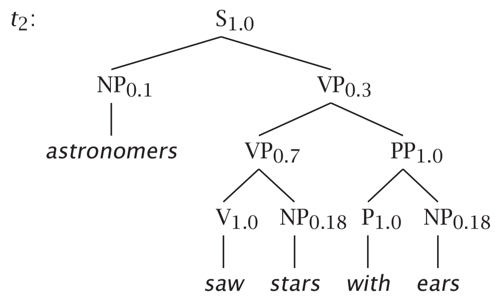
A simple PCFG (in CNF)

S → NP VP	1.0	NP → NP PP	0.4
PP → P NP	1.0	NP → *astronomers*	0.1
VP → V NP	0.7	NP → *ears*	0.18
VP → VP PP	0.3	NP → *saw*	0.04
P → *with*	1.0	NP → *stars*	0.18
V → *saw*	1.0	NP → *telescopes*	0.1

t_1:



t_2:



The two parse trees' probabilities and the sentence probability

$$\begin{aligned}
P(t_1) &= 1.0 \times 0.1 \times 0.7 \times 1.0 \times 0.4 \\
&\qquad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0009072 \\
P(t_2) &= 1.0 \times 0.1 \times 0.3 \times 0.7 \times 1.0 \\
&\qquad \times 0.18 \times 1.0 \times 1.0 \times 0.18 \\
&= 0.0006804 \\
P(w_{15}) &= P(t_1) + P(t_2) = 0.0015876
\end{aligned}$$

Attachment ambiguities: A key parsing decision

- The main problem in parsing is working out how to 'attach' various kinds of constituents – PPs, adverbial or participial phrases, coordinations, and so on
- Prepositional phrase attachment
  - *I saw the man with a telescope*
- What does *with a telescope* modify?
  - The verb *saw*?
  - The noun *man*?
- Is the problem 'AI-complete'? Yes, but . . .

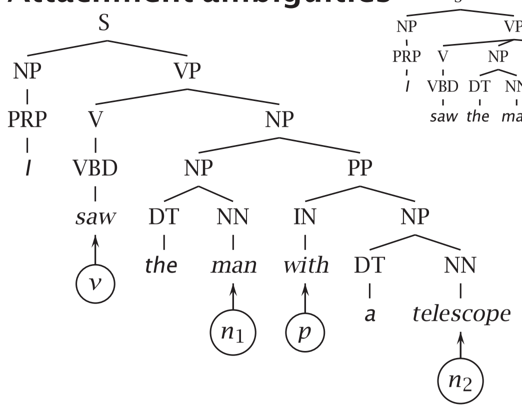
Attachment ambiguities (2)

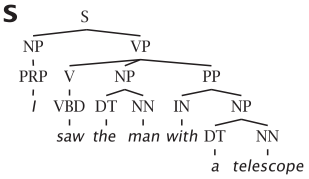
- Proposed simple structural factors
  - Right association (Kimball 1973) = 'low' or 'near' attachment = 'late closure' (of NP) [NP → NP PP]
  - Minimal attachment (Frazier 1978) [depends on grammar] = 'high' or 'distant' attachment = 'early closure' (of NP) [VP → V NP PP]
- Such simple structural factors dominated in early psycholinguistics, and are still widely invoked.
- In the V NP PP context, right attachment gets it right in 55–67% of cases.
- But that means it gets it wrong in 33–45% of cases

Importance of lexical factors

- Words are good predictors (or even inducers) of attachment (even absent understanding):
  - The children ate the cake with a spoon.
  - The children ate the cake with frosting.
  - Moscow sent more than 100,000 soldiers into Afghanistan
  - Sydney Water breached an agreement with NSW Health
- Ford et al. (1982):
  - Ordering is jointly determined by strengths of alternative lexical forms, alternative syntactic rewrite rules, and the sequence of hypotheses in parsing

Attachment ambiguities





Assumptions of PCFGs

1. Place invariance (like time invariance in HMM):

$$\forall k \quad P(N^j_{k(k+c)} \to \zeta) \text{is the same}$$

2. Context-free:

$$P(N^j_{kl} \to \zeta | \text{words outside } w_k \dots w_l) = P(N^j_{kl} \to \zeta)$$

3. Ancestor-free:

$$P(N^j_{kl} \to \zeta | \text{ancestor nodes of } N^j_{kl}) = P(N^j_{kl} \to \zeta)$$

The sufficient statistics of a PCFG are thus simply counts of how often different local tree configurations occurred
(= counts of which grammar rules were applied).

Some features of PCFGs

Reasons to use a PCFG, and some idea of their limitations:

- Partial solution for grammar ambiguity: a PCFG gives some idea of the plausibility of a sentence.
- But, in the simple case, not a very good idea, as independence assumptions are two strong (e.g., not lexicalized).
- Gives a probabilistic language model for English.
- In the simple case, a PCFG is a worse language model for English than a trigram model.
- Better for grammar induction (Gold 1967 vs. Horning 1969)
- Robustness. (Admit everything with low probability.)

Some features of PCFGs

- A PCFG encodes certain biases, e.g., that smaller trees are normally more probable.
- One can hope to combine the strengths of a PCFG and a trigram model.

We'll look at simple PCFGs first. They have certain inadequacies, but we'll see that most of the state-of-the-art probabilistic parsers are fundamentally PCFG models, just with various enrichments to the grammar

Questions for PCFGs

Just as for HMMs, there are three basic questions we wish to answer:

- Language modeling: $P(w_{1m}|G)$
- Parsing: $\arg\max_t P(t|w_{1m}, G)$
- Learning algorithm: Find G such that $P(w_{1m}|G)$ is maximized.

Chomsky Normal Form grammars

We'll do the case of Chomsky Normal Form grammars, which only have rules of the form:

$$\begin{aligned} N^i &\to N^j N^k \\ N^i &\to w^j \end{aligned}$$

Any CFG can be represented by a weakly equivalent CFG in Chomsky Normal Form. It's straightforward to generalize the algorithm (recall chart parsing).

Probabilistic Regular Grammar:

$N^i \to w^j N^k$
$N^i \to w^j$
Start state, N^1

HMM:

$$\sum_{w_{1n}} P(w_{1n}) = 1 \quad \forall n$$

whereas in a PCFG or a PRG:

$$\sum_{w \in L} P(w) = 1$$

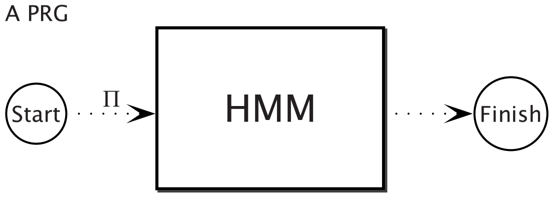
Probabilistic Regular Grammar

Consider:

$$P(\textit{John decided to bake a})$$

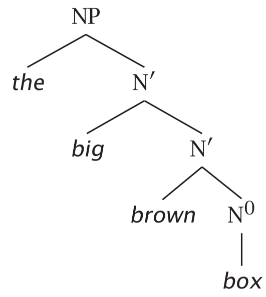
High probability in HMM, low probability in a PRG or a PCFG. Implement via sink (end) state.

A PRG



Comparison of HMMs (PRGs) and PCFGs

$$\begin{array}{lccccccccc} X: & \text{NP} & \longrightarrow & \text{N}' & \longrightarrow & \text{N}' & \longrightarrow & \text{N}^0 & \longrightarrow & \text{sink} \\ & | & & | & & | & & | & & \\ O: & \textit{the} & & \textit{big} & & \textit{brown} & & \textit{box} & & \end{array}$$



Inside and outside probabilities

This suggests: whereas for an HMM we have:

Forwards = $\alpha_i(t) = P(w_{1(t-1)}, X_t = i)$

Backwards = $\beta_i(t) = P(w_{tT}|X_t = i)$

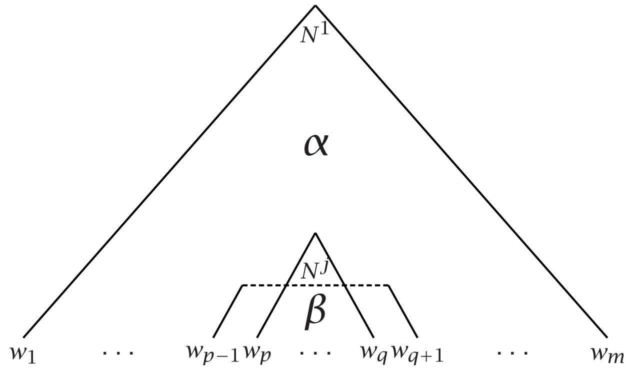
for a PCFG we make use of Inside and Outside probabilities, defined as follows:

Outside = $\alpha_j(p,q) = P(w_{1(p-1)}, N^j_{pq}, w_{(q+1)m}|G)$

Inside = $\beta_j(p,q) = P(w_{pq}|N^j_{pq}, G)$

A slight generalization of dynamic Bayes Nets covers PCFG inference by the inside-outside algorithm (and-or tree of conjunctive daughters disjunctively chosen)

Inside and outside probabilities in PCFGs.



Probability of a string

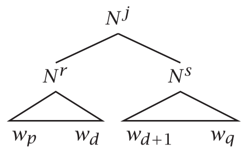
Inside probability

$$\begin{aligned} P(w_{1m}|G) &= P(N^1 \Rightarrow w_{1m}|G) \\ &= P(w_{1m}, N^1_{1m}, G) \quad = \beta_1(1,m) \end{aligned}$$

Base case: We want to find $\beta_j(k,k)$ (the probability of a rule $N^j \to w_k$):

$$\begin{aligned} \beta_j(k,k) &= P(w_k|N^j_{kk}, G) \\ &= P(N^j \to w_k|G) \end{aligned}$$

Induction: We want to find $\beta_j(p,q)$, for $p < q$. As this is the inductive step using a Chomsky Normal Form grammar, the first rule must be of the form $N^j \to N^r \quad N^s$, so we can proceed by induction, dividing the string in two in various places and summing the result:



These inside probabilities can be calculated bottom up.

For all j,

$$
\begin{aligned}
\beta_j(p,q) &= P(w_{pq}|N^j_{pq},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(w_{pd},N^r_{pd},w_{(d+1)q},N^s_{(d+1)q}|N^j_{pq},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{pd},N^s_{(d+1)q}|N^j_{pq},G) \\
&\qquad P(w_{pd}|N^j_{pq},N^r_{pd},N^s_{(d+1)q},G) \\
&\qquad P(w_{(d+1)q}|N^j_{pq},N^r_{pd},N^s_{(d+1)q},w_{pd},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^r_{pd},N^s_{(d+1)q}|N^j_{pq},G) \\
&\qquad P(w_{pd}|N^r_{pd},G)P(w_{(d+1)q}|N^s_{(d+1)q},G) \\
&= \sum_{r,s}\sum_{d=p}^{q-1} P(N^j \to N^r N^s)\beta_r(p,d)\beta_s(d+1,q)
\end{aligned}
$$

Calculation of inside probabilities (CKY algorithm)

	1	2	3	4	5
1	$\beta_{\text{NP}} =$ 0.1		$\beta_{\text{S}} =$ 0.0126		$\beta_{\text{S}} =$ 0.0015876
2		$\beta_{\text{NP}} =$ 0.04 $\beta_{\text{V}} =$ 1.0	$\beta_{\text{VP}} =$ 0.126		$\beta_{\text{VP}} =$ 0.015876
3			$\beta_{\text{NP}} =$ 0.18		$\beta_{\text{NP}} =$ 0.01296
4				$\beta_{\text{P}} =$ 1.0	$\beta_{\text{PP}} =$ 0.18
5					$\beta_{\text{NP}} =$ 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Outside probabilities

Probability of a string: For any k, $1 \le k \le m$,

$$
\begin{aligned}
P(w_{1m}|G) &= \sum_j P(w_{1(k-1)},w_k,w_{(k+1)m},N^j_{kk}|G) \\
&= \sum_j P(w_{1(k-1)},N^j_{kk},w_{(k+1)m}|G) \\
&\qquad \times P(w_k|w_{1(k-1)},N^j_{kk},w_{(k+1)n},G) \\
&= \sum_j \alpha_j(k,k)P(N^j \to w_k)
\end{aligned}
$$

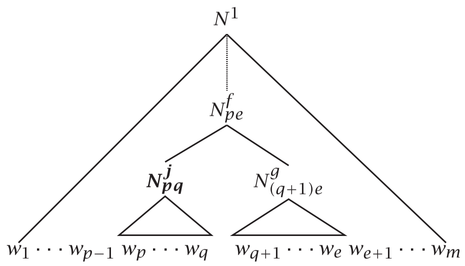
Inductive (DP) calculation: One calculates the outside probabilities top down (after determining the inside probabilities).

Outside probabilities

Base Case:

$\alpha_1(1,m) = 1$
$\alpha_j(1,m) = 0$, for $j \ne 1$

Inductive Case: it's either a left or right branch – we will some over both possibilities and calculate using outside *and* inside probabilities



Inductive Case:

$$
\begin{aligned}
\alpha_j(p,q) &= \Big[\sum_{f,g}\sum_{e=q+1}^{m} P(w_{1(p-1)},w_{(q+1)m},N^f_{pe},N^j_{pq},N^g_{(q+1)e})\Big] \\
&\quad + \Big[\sum_{f,g}\sum_{e=1}^{p-1} P(w_{1(p-1)},w_{(q+1)m},N^f_{eq},N^g_{e(p-1)},N^j_{pq})\Big] \\
&= \Big[\sum_{f,g}\sum_{e=q+1}^{m} P(w_{1(p-1)},w_{(e+1)m},N^f_{pe})P(N^j_{pq},N^g_{(q+1)e}|N^f_{pe}) \\
&\quad \times P(w_{(q+1)e}|N^g_{(q+1)e})\Big] + \Big[\sum_{f,g}\sum_{e=1}^{p-1} P(w_{1(e-1)},w_{(q+1)m},N^f_{eq}) \\
&\quad \times P(N^g_{e(p-1)},N^j_{pq}|N^f_{eq})P(w_{e(p-1)}|N^g_{e(p-1)})\Big] \\
&= \Big[\sum_{f,g}\sum_{e=q+1}^{m} \alpha_f(p,e)P(N^f \to N^j\, N^g)\beta_g(q+1,e)\Big] \\
&\quad + \Big[\sum_{f,g}\sum_{e=1}^{p-1} \alpha_f(e,q)P(N^f \to N^g\, N^j)\beta_g(e,p-1)\Big]
\end{aligned}
$$

Overall probability of a node existing

As with a HMM, we can form a product of the inside and outside probabilities. This time:

$$
\begin{aligned}
&\alpha_j(p,q)\beta_j(p,q) \\
&= P(w_{1(p-1)},N^j_{pq},w_{(q+1)m}|G)P(w_{pq}|N^j_{pq},G) \\
&= P(w_{1m},N^j_{pq}|G)
\end{aligned}
$$

Therefore,

$$p(w_{1m},N_{pq}|G) = \sum_j \alpha_j(p,q)\beta_j(p,q)$$

Just in the cases of the root node and the preterminals, we know there will always be some such constituent.

Finding the most likely parse (Viterbi algorithm)

Like inside algorithm, but find maximum rather than sum

Record which rule gave this maximum

$\delta_i(p,q)$ = the highest inside probability parse of a subtree N^i_{pq}

1. Initialization: $\delta_i(p,p) = P(N^i \to w_p)$
2. Induction
$$\delta_i(p,q) = \max_{\substack{1\le j,k\le n\\ p\le r<q}} P(N^i \to N^j\ N^k)\delta_j(p,r)\delta_k(r+1,q)$$
3. Store backtrace
$$\psi_i(p,q) = \underset{(j,k,r)}{\arg\max}\, P(N^i \to N^j\ N^k)\delta_j(p,r)\delta_k(r+1,q)$$
4. From start symbol N^1, most likely parse t is:
t begins with $\psi_1(1,m)$. $\quad P(\hat{t}) = \delta_1(1,m)$

Calculation of Viterbi probabilities (CKY algorithm)

	1	2	3	4	5
1	$\delta_{NP} =$ 0.1		$\delta_S =$ 0.0126		$\delta_S =$ 0.0009072
2		$\delta_{NP} =$ 0.04 $\delta_V =$ 1.0	$\delta_{VP} =$ 0.126		$\delta_{VP} =$ 0.009072
3			$\delta_{NP} =$ 0.18		$\delta_{NP} =$ 0.01296
4				$\delta_P =$ 1.0	$\delta_{PP} =$ 0.18
5					$\delta_{NP} =$ 0.18
	astronomers	*saw*	*stars*	*with*	*ears*

Learning PCFGs (1)

- We would like to calculate how often each rule is used:
$$\hat{P}(N^j \to \zeta) = \frac{C(N^j \to \zeta)}{\sum_\gamma C(N^j \to \gamma)}$$
- If we have labeled data, we count and find out
- Relative frequency again gives maximum likelihood probability estimates
- This is the motivation for building *Treebanks* of hand-parsed sentences

Learning PCFGs (2): the Inside-Outside algorithm (Baker 1979)

- Otherwise we work iteratively from expectations of current model.
- We construct an EM training algorithm, as for HMMs
- For each sentence, at each iteration, we work out expectation of how often each rule is used using inside and outside probabilities
- We assume sentences are independent and sum expectations over parses of each
- We re-estimate rules based on these ‘counts’