Parsing

Treebank Grammars

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2019

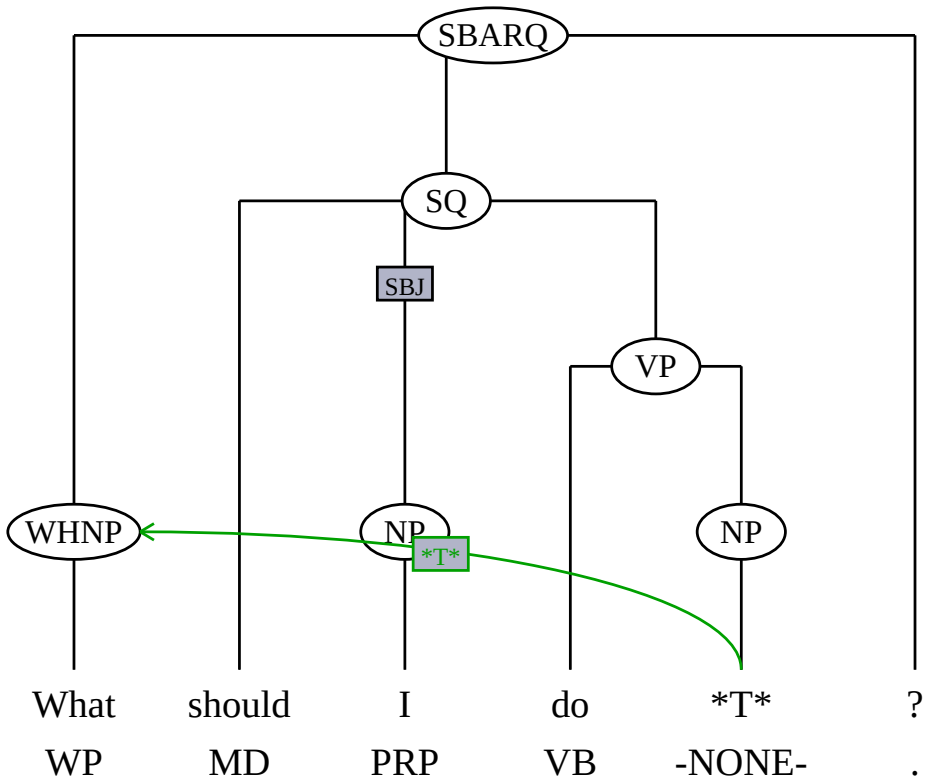
Table of contents

Treebanks (1)

- Treebanks are corpora (i.e., collections of texts) where each sentence is annotated with a syntactic structure.
- The syntactic structure can be a **constituency structure** or a **dependency structure**.
- Constituency-based data driven parsing is usually done by learning a grammar (in most cases a PCFG) from a constituency treebank and using this grammar for parsing.
- Dependency-based data driven parsing is usually done by learning a dependency parser (e.g., a classifier) from the treebank.

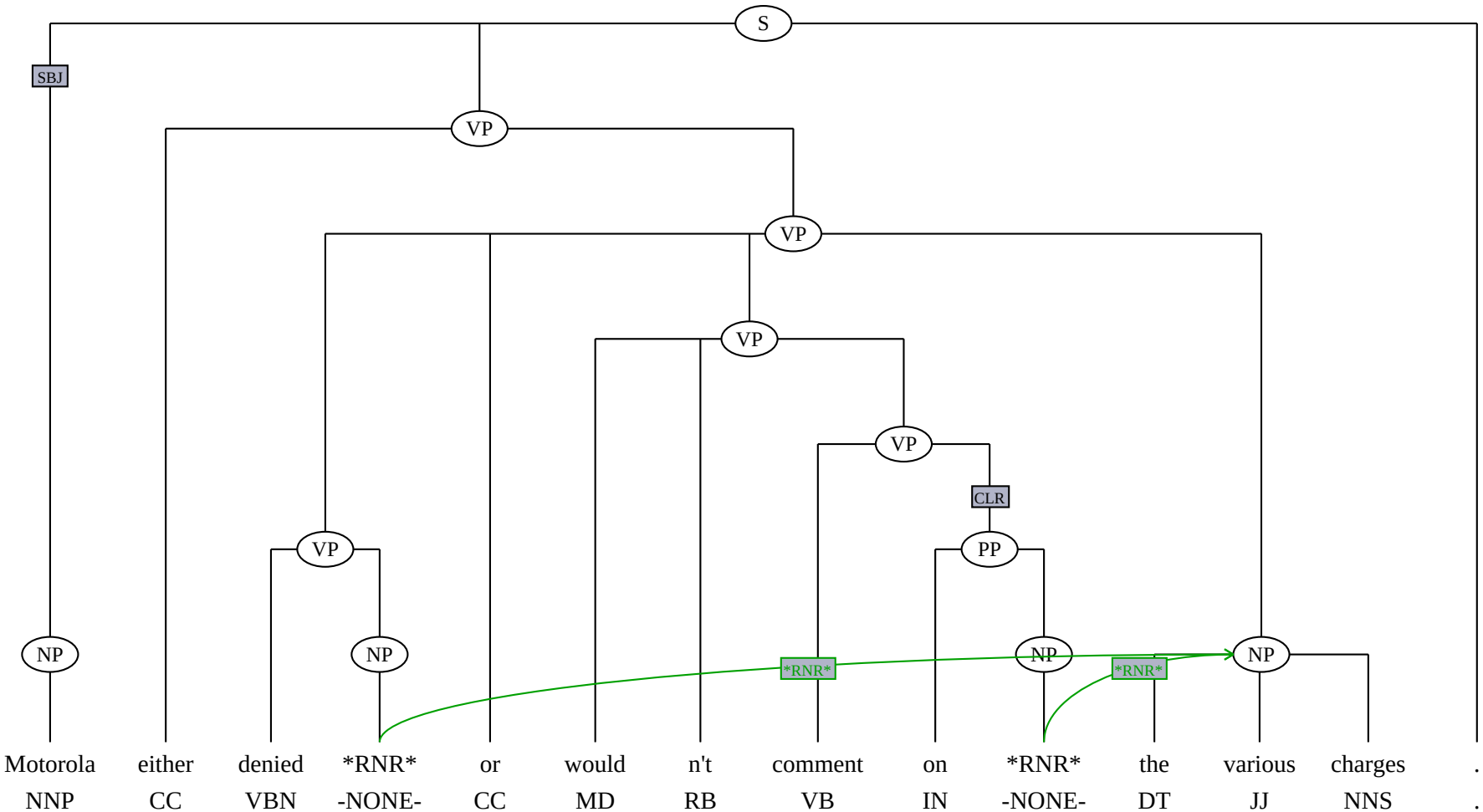
Treebanks (2)

Sample tree from the Penn Treebank (PTB, Marcus et al., 1993):



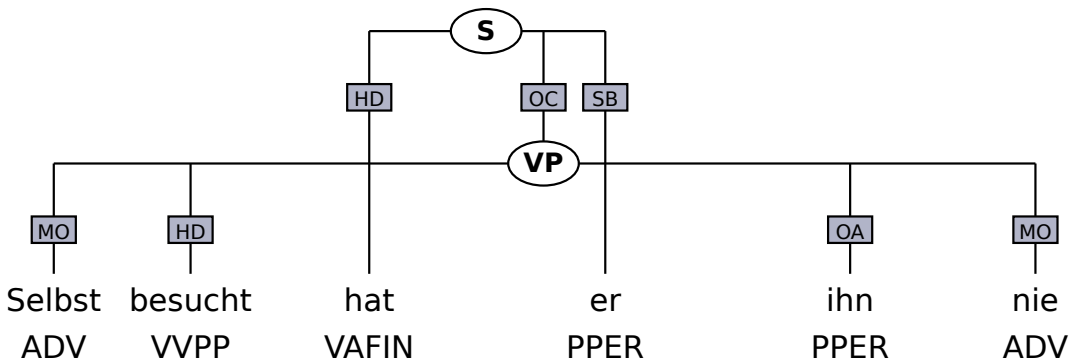
Treebanks (3)

A further example from the Penn Treebank:



Treebanks (3)

A sample tree from Negra, a German treebank (Skut et al., 1997):



Grammar Extraction (1)

Having a treebank, a simple way to extract a latent PCFG is the following:

- We first do some preprocessing (removal of traces, of crossing branches, ...).
- Then, we binarize the trees, i.e., we make sure all rhight-hand sides have length 2.
- For all $A \to \alpha \in P$, the estimated probability $p(A \to \alpha)$ is

$$p(A \to \alpha) = \frac{count(A \to \alpha)}{count(A)}$$

  where $count(A \to \alpha)$ is the number of occurrences of the production in the treebank and $count(A)$ the number of A-nodes in the treebank.

This is called a Maximum Likelihood Estimator.

Grammar Extraction (2)

Problem with such grammars: Independence assumptions are too strong.
Therefore, a series of techniques for grammar refinement have been proposed:

- **Lexicalization** of PCFGs (Collins, 2003)
- **Markovization**: Instead of using unique new non-terminals during binarization, we always use the same X, attaching some vertical and horizontal context to it (Klein & Manning, 2003)
- **Category splitting and merging**: whenever a single category A behaves differently in different context, we split it into several new categories, depending on context. This can be done automatically (Petrov et al., 2006)

Markovization (1)

The Chomsky Normal Form (CNF) binarization introduced earlier in the course is such that for every rule $A_0 \rightarrow A_1 \ldots A_n$ with $n > 2$, $n - 1$ new nonterminals are introduced:

$$A_0 \rightarrow A_1 X_1, X_1 \rightarrow A_2 X_2, \ldots, X_{n-1} \rightarrow A_{n-1} A_n$$

Problems:

- Too many non-terminals.
- Lack of generalization (non-terminals are highly sepcialized).

For this reason, we introduce **markovization** (Klein & Manning, 2003).

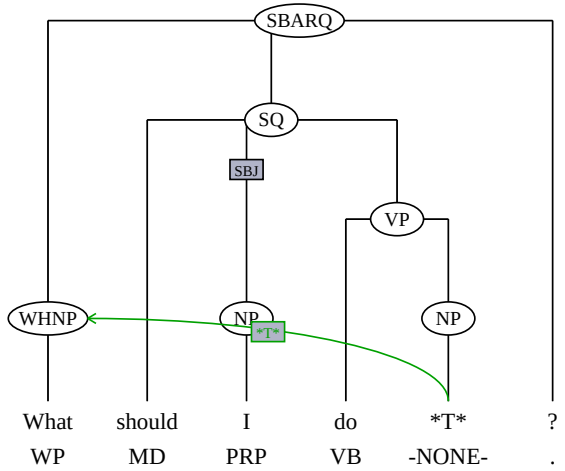
Markovization (2)

Idea:

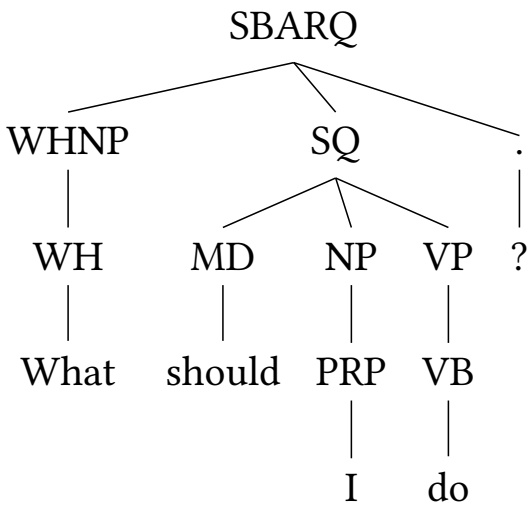
- Introduce only a **single new non-terminal** for the new rules introduced during binarization.
- add **vertical and horizontal context** from the original trees to each occurrence of this new non-terminal.
  - As vertical context, we add the first v labels on the path from the root node of the tree that we want to binarize to the root of the entire treebank tree. The vertical context is collected during grammar extraction and then taken into account during binarization of the rules.
  - As horizontal context, during binarization of a rule $A \rightarrow A_0 \ldots A_n$, for the new non-terminal that comprises the right-hand side elements $A_i \ldots A_n$ (for some $1 \leq i \leq n$), we add the first h elements of $A_i, A_{i-1}, \ldots, A_0$.

Markovization (3)

Example: Consider again PTP tree from slide 4:

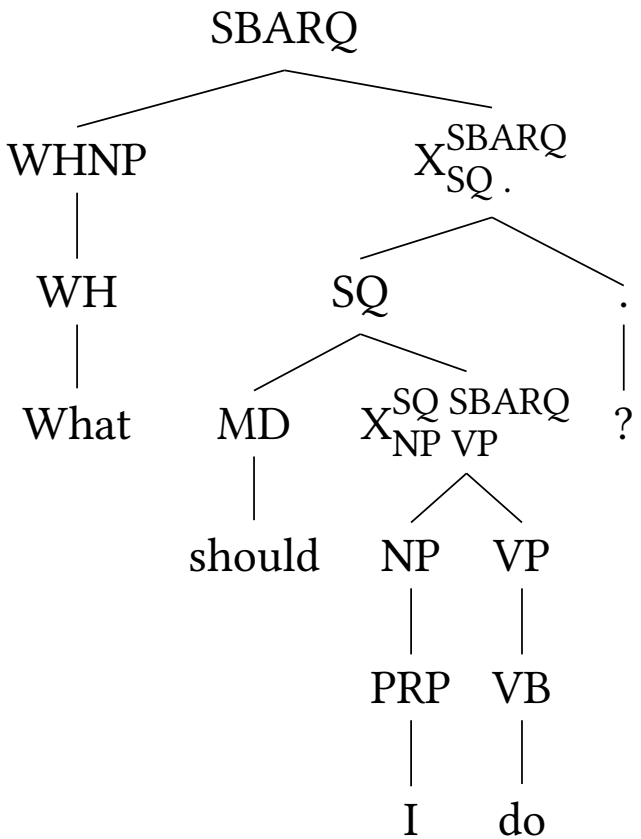


Removal of the trace leads to



Markovization (4)

Binarization and Markovization with $v = 2$ and $h = 2$ (superscript = vertical context, subscript = horizontal context of the new non-terminal X):

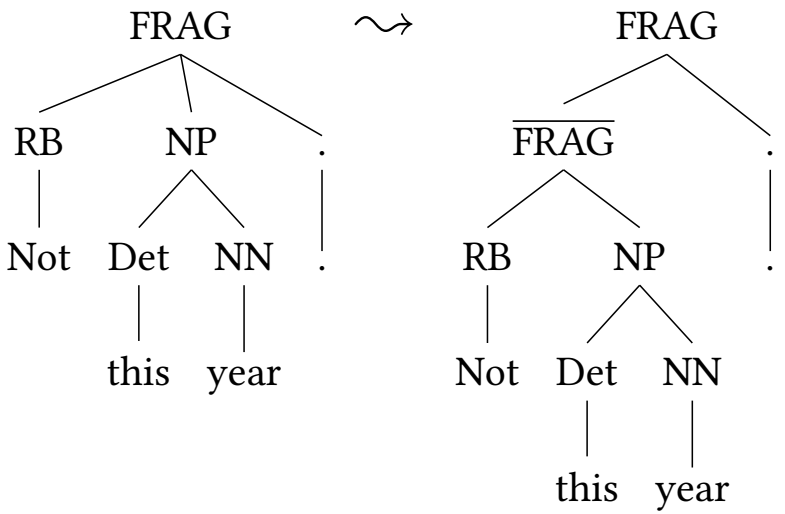


(oftentimes, $v = 1$ and $h = 2$ are chosen)

Latent categories (1)

There also have been a range of approaches to refining treebank grammars. One of the best performing approaches is the Berkeley parser Petrov et al. (2006).

- Starting point: Penn Treebank trees.
- Binarization of these trees: left-branching binarization with new intermediate non-terminals $\overline{A}$ where A is the root of the tree one wants to binarize.



- Start with the treebank grammar obtained from these trees.

Latent categories (2)

Iteration for learning new labels and estimating new probabilities:

Given a current PCFG, repeat the following until no more successful splits can be found:

- Split non-terminals:
  - Split every non-terminal A into 2 new symbols A_1, A_2(e.g, NP $\rightsquigarrow$ NP_1, NP_2),
  - replace every rule $A \rightarrow \alpha$ with $A_1 \rightarrow \alpha$ and $A_2 \rightarrow \alpha$, both with the same probability as the original rule,
  - and for every occurrence of A in a righthand side of some $B \rightarrow \gamma$, replace $B \rightarrow \gamma$ with two new rules, one with A_1 instead of A and another one with A_2, dividing the probability of the original rule between these two new rules. This is done repeatedly until all old non-terminals have been removed.
  - Furthermore, add some small amount of randomness to the probabilities to break the symmetry.

Latent categories (3)

- Then, probabilities are re-estimating using the inside-outside EM algorithm (Collins) with the split grammar, based on the correct treebank bracketing and the coarser node labels in the treebank.
- Each split that has contributed to increasing the probability of the training data is kept and the other splits are reversed.

Result:

- POS tags get split, for instance VBZ is split into 11 different POS tags.
- Phrasal non-terminals get split as well, for example different VP categories for infinite VPs, passive VPs, intransitive VPs etc.
- The best evaluation was obtained with a resulting grammar with 1043 symbols, F_1 score of 90.2% on the Penn treebank.

EM Training

Supervised data-driven PCFG parsing: Given a treebank, read off the rules and estimate their probabilities based on the counts of the rules.

EM Training

Supervised data-driven PCFG parsing: Given a treebank, read off the rules and estimate their probabilities based on the counts of the rules.

More challenging: Unsupervised parameter estimation: Given a CFG and unannotated training data, estimate the probabilities of the rules.

We use the EM algorithm, based on the inside and outside computations from the previous slides.

EM Training

Underlying ideas, as in the HMM parameter estimation:

- We estimate parameters iteratively: we start with some parameters and use the estimated probabilities to derive better and better parameters.
- We use our current parameters to estimate (fractional) counts of possible parse trees and possible rules. In other words, the probability mass assigned to the training corpus gets distributed among the possible parse trees.
- These fractional counts are then used to compute the parameters of the next model.

EM Training

For each rule $r = A \to \gamma \in P$, we start with some initial probabilities $p^{(0)}(r)$ that can be chosen randomly. In each iteration, based on the probabilities $p^{(i)}$, new probabilities $p^{(i+1)}$ are estimated.

Intuition:

$$p^{(i+1)}(A \to \gamma) = \frac{\text{expected count of } A \to \gamma}{\text{expected count of non-terminal } A}$$

more precisely

$$p^{(i+1)}(A \to \gamma) = \frac{f^{(i)}(A \to \gamma)}{\sum_{A \to \gamma' \in P} f^{(i)}(A \to \gamma')}$$

In the E-step of the algorithm, we compute the fractional counts $f^{(i)}(r)$ for all $r \in P$ and in the M-step, we re-estimate the probabilities according to these new counts.

EM Training

We can think of this as follows:

- Our training data are sentences $w(1), \ldots, w(N)$.
- In each iteration, based on the current probabilities, we create a treebank for training:

  For each of the sentences, the treebank contains all possible parse trees. But tree t does not occur once in the treebank, instead, it occurs $P(t)$ times.
- Consequently, when counting occurrences of rules in the treebank in order to estimate new probabilities, an occurrence of some rule r in a parse tree t does not add 1 to the count but it adds $P(t)$.
- The resulting count for rule r, summing up the probabilities of the parse trees for every occurrence of r, is then the exptected count of r.

EM Training

Computation of the fractional counts for a single sentence w: We distribute $P(w)$ among all the rules used in any of the parse trees of w, in accordance with the probability of these parse trees.

We have

$$P(w) \quad = \quad \alpha_{S,1,|w|}$$

and

$$\begin{aligned} P(S \;&\stackrel{*}{\Rightarrow}\; w_1 \ldots w_{i-1} A w_{j+1} \ldots w_{|w|} \\ &\Rightarrow\; w_1 \ldots w_{i-1} B C w_{j+1} \ldots w_{|w|} \\ &\stackrel{*}{\Rightarrow}\; w_1 \ldots w_{k-1} C w_{j+1} \ldots w_{|w|} \\ &\stackrel{*}{\Rightarrow}\; w_1 \ldots w_{|w|}) \\ &= \beta_{A,i,j} \alpha_{B,i,k-1} \alpha_{C,k,j} p(A \to BC) \end{aligned}$$

EM Training

Computation of $C_w(A \to \gamma)$ for a sentence w

Let $G = \langle N, T, P, S \rangle$ be a PCFG with probabilities $p(r)$ for all rules $r \in P$ and let $w \in T^*$ be an input sentence.

1. Calculate the inside and outside probabilities $\alpha_{A,i,j}$ and $\beta_{A,i,j}$ for all $A \in N$ and $1 \leq i < j \leq |w|$.
2. For every rule of the form $A \to BC$:

$$C_w(A \to BC) = \sum_{1 \leq i < k \leq j \leq n} \frac{\beta_{A,i,j} \alpha_{B,i,k-1} \alpha_{C,k,j} p(A \to BC)}{\alpha_{S,1,|w|}}$$

3. For every rule of the form $A \to a$:

$$C_w(A \to a) = \sum_{1 \leq i \leq n, w_i = a} \frac{\beta_{A,i,i} p(A \to a)}{\alpha_{S,1,|w|}}$$

EM Training

E-step of EM parameter estimation

$G = \langle \{S, A, X\}, \{a, c\}, P, S, p \rangle$ with P and p as follows:

0.5: S $\rightarrow$ AX 0.3: S $\rightarrow$ XA 0.2: S $\rightarrow$ c 1: X $\rightarrow$ AA 1: A $\rightarrow$ a

Training data is just the single sentence *aaa*.

Inside values:

aaa:				
j				
3	(0.8,S)	(1,X)	(1,A)	
2	(1,X)	(1,A)		
1	(1,A)			
	1	2	3	*i*

Outside values:

j				
3	(1,S)	(0.5,X)	(0.8,A)	
2	(0.3,X)	(0.8,A)		
1	(0.8,A)			
	1	2	3	*i*

EM Training

E-step of EM parameter estimation continued

E-step: Compute the new counts $C_{aaa}(S \to AX)$, $C_{aaa}(S \to XA)$ and $C_{aaa}(S \to c)$ and, based on these, the new frequencies $f(S \to AX)$, $f(S \to XA)$ and $f(S \to c)$.

$$C_{aaa}(S \to AX) = \frac{0.5 \cdot \alpha_{A,1,1} \cdot \alpha_{X,2,3}}{0.8} = \frac{0.5}{0.8} = \frac{5}{8}$$
$$C_{aaa}(S \to XA) = \frac{0.3 \cdot \alpha_{X,1,2} \cdot \alpha_{A,3,3}}{0.8} = \frac{0.3}{0.8} = \frac{3}{8}$$
$$C_{aaa}(S \to c) = 0$$

Since we have only one training sentence, we obtain
$f(S \to AX) = \frac{5}{8}, f(S \to XA) = \frac{3}{8}, f(S \to c) = 0$

M-step: Compute the new probabilities $\hat{p}(S \to AX)$, $\hat{p}(S \to XA)$ and $\hat{p}(S \to c)$ that arise after the first iteration.

$$\hat{p}(S \to AX) = \frac{\frac{5}{8}}{\frac{3+5+0}{8}} = \frac{5}{8}, \hat{p}(S \to XA) = \frac{\frac{3}{8}}{\frac{3+5+0}{8}} = \frac{3}{8}, \hat{p}(S \to c) = 0$$

EM Training

In order to calculate the fractional count $f^{(i)}(A \to \gamma)$, sum over the counts $C_w(A \to \gamma)$ for all sentences in the training corpus:

E-step

Let our training corpus consist of sentences $w^{(1)} \ldots w^{(N)}$ and let the PCFG and its probability function p be as above.

$$f(A \to \gamma) = \sum_{1 \leq m \leq N} C_{w^{(m)}}(A \to \gamma)$$

This is the **E (expectation)** step for our parameters.

EM Training

From these frequencies (= fractional counts) $f(A \to \gamma)$, we can estimate new rule probabilities $\hat{p}$ towards maximizing the observed data:

M-step

For every $A \to \gamma \in P$:

$$\hat{p}(A \to \gamma) = \frac{f(A \to \gamma)}{\sum_{A \to \gamma' \in P} f(A \to \gamma')}$$

This is the **M (maximization)** step for the rule probabilities.

EM Training

EM algorithm for estimation of p for a PCFG; training corpus is a sequence of sentences $w^{(1)}, \ldots, w^{(N)}$

initialize p

iterate until convergence:

E-step

for every $1 \leq m \leq N$: compute $C_w(A \to \gamma)$ as above

for every $r \in P$: $f(A \to \gamma) = \sum_{1 \leq m \leq N} C_{w^{(m)}}(A \to \gamma)$

M-step

for every $A \to \gamma \in P$: $\hat{p}(A \to \gamma) = \frac{f(A \to \gamma)}{\sum_{A \to \gamma' \in P} f(A \to \gamma')}$

return p

Treebank refinement

- So far, we have seen completely unsupervised training. Each iteration has a complexity of $\mathcal{O}(|N|^3|w|^3)$.
- The complexity decreases considerably if we know the parse trees of the sentences except for the node labels. I.e., we know about the bracketing of the trees.
- Pereira & Schabes (1992) show how this information can be integrated into the inside outside computation:
  1. Given a sentence w with its bracketing, we define $\overline{c}(i,j)$ as 1 if a subtree spanning $w_i \ldots w_j$ exists and otherwise it is 0.
  2. For every value $\alpha_{A,i,j}$ and $\beta_{A,i,j}$, we multiply with the factor $\overline{c}(i,j)$. Consequently all values where there is no corresponding bracketing are set to 0.
- Inside outside computation becomes linear in the size of the input.

Collins, Michael. ???? The inside-outside algorithm. www.cs.columbia.edu/~mcollins/io.pdf.

Collins, Michael. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational Linguistics* 29(4). 589–637.

Klein, Dan & Christopher D. Manning. 2003. Fast exact inference with a factored model for Natural Language parsing. In *Advances in neural information processing systems 15 (nips)*, 3–10. Vancouver, BC: MIT Press.

Marcus, Mitchell P., Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2). 313–330. Special Issue on Using Large Corpora: II.

Pereira, Fernando & Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the 30th annual meeting of the association for computational linguistics*, 128–135. Newark, Delaware, USA: Association for Computational Linguistics. doi:10.3115/981967.981984. http://www.aclweb.org/anthology/P92-1017.

Petrov, Slav, Leon Barrett, Romain Thibaux & Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the acl*, 433–440. Sydney.

Skut, Wojciech, Brigitte Krenn, Thorsten Brants & Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th applied natural language processing conference*, 88–95. Washington, DC.