1584th EM iteration. The estimate which is calculated here is

$p_{1584,1}(x_1)$	x_1
0.158396	1
0.141282	2
0.204291	3
0.0785532	4
0.172207	5
0.24527	6

$p_{1584,2}(x_2)$	x_2
0.239281	1
0.260559	2
0.104026	3
0.111957	4
0.134419	5
0.149758	6

yielding

$p_{1584}(x_1, x_2)$	$x_2 = 1$	$x_2 = 2$	$x_2 = 3$	$x_2 = 4$	$x_2 = 5$	$x_2 = 6$
$x_1 = 1$	0.0379012	0.0412715	0.0164773	0.0177336	0.0212914	0.0237211
$x_1 = 2$	0.0338061	0.0368123	0.014697	0.0158175	0.018991	0.0211581
$x_1 = 3$	0.048883	0.0532299	0.0212516	0.0228718	0.0274606	0.0305942
$x_1 = 4$	0.0187963	0.0204678	0.00817158	0.00879459	0.0105591	0.011764
$x_1 = 5$	0.0412059	0.0448701	0.017914	0.0192798	0.0231479	0.0257894
$x_1 = 6$	0.0586885	0.0639074	0.0255145	0.0274597	0.032969	0.0367312

In this example, more EM iterations will result in exactly the same re-estimates. So, this is a strong reason to quit the EM procedure. Comparing $p_{1584,1}$ and $p_{1584,2}$ with the results of Example 3 *(Hint: where we have assumed that a complete-data corpus is given to us!)*, we see that the EM algorithm yields pretty similar estimates.

5 Probabilistic Context-Free Grammars

This Section provides a more substantial example based on the **context-free grammar** or **CFG** formalism, and it is organized as follows: First, we will give some background information about CFGs, thereby motivating that treating CFGs as generators leads quite naturally to the notion of a probabilistic context-free grammar (PCFG). Second, we provide some additional background information about ambiguity resolution by probabilistic CFGs, thereby focusing on the fact that probabilistic CFGs can resolve ambiguities, if the underlying CFG has a sufficiently high expressive power. For other cases, we are pin-pointing to some useful grammar-transformation techniques. Third, we will investigate the standard probability model of CFGs, thereby proving that this model is restricted in almost all cases of interest. Furthermore, we will give a new formal proof that maximum-likelihood estimation of a CFG's probability model on a corpus of trees is equal to the well-known and especially simple treebank-training method. Finally, we will present the EM algorithm for training a (manually written) CFG on a corpus of sentences, thereby pin-pointing to the fact that EM training simply consists of an iterative sequence of treebank-training steps. Small toy examples will accompany all proofs that are given in this Section.

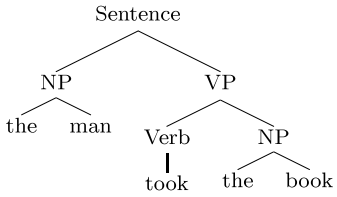
Background: Probabilistic Modeling of CFGs

Being a bit sloppy (see e.g. Hopcroft and Ullman (1979) for a formal definition), a CFG simply consists of a finite set of **rules**, where in turn, each rule consists of two parts being

separated by a special symbol " $\longrightarrow$ ", the so-called **rewriting symbol**. The two parts of a rule are made up of so-called **terminal** and **non-terminal** symbols: a rule's left-hand side simply consists of a single non-terminal symbol, whereas the right-hand side is a finite sequence of terminal and non-terminal symbols[7]. Finally, the set of non-terminal symbols contains at least one so-called **starting symbol**. CFGs are also called **phrase-structure grammars**, and the formalism is equivalent to **Backus-Naur forms** or **BNF** introduced by Backus (1959). In computational linguistics, a CFG is usually used in two ways

- as a **generator**: a device for generating sentences, or
- as a **parser**: a device for assigning structure to a given sentence

In the following, we will briefly discuss these two issues. First of all, note that in natural language, words do not occur in any order. Instead, languages have constraints on word order[8]. The central idea underlying phrase-structure grammars is that words are organized into **phrases**, i.e., grouping of words that form a unit. Phrases can be detected, for example, by their ability (i) to stand alone (e.g. as an answer of a wh-question), (ii) to occur in various sentence positions, or by their ability (iii) to show uniform syntactic possibilities for expansion or substitution. As an example, here is the very first context-free grammar parse tree presented by Chomsky (1956):



As being displayed, Chomsky identified for the sentence "the man took the book" (encoded in the leaf nodes of the parse tree) the following phrases: two **noun phrases**, "the man" and "the book" (the figure displays them as NP subtrees), and one **verb phrase**, "took the book" (displayed as VP subtree). The following list of sentences, where these three phrases have been substituted or expanded, bears some evidence for Chomsky's analysis:

$$\left\{\begin{array}{c}\text{he}\\ \text{the man}\\ \text{the tall man}\\ \text{the very tall man}\\ \text{the tall man with sad eyes}\end{array}\right\}\ \text{took}\ \left\{\begin{array}{c}\text{it}\\ \text{the book}\\ \text{the interesting book}\\ \text{the very interesting book}\\ \text{the very interesting book with 934 pages}\end{array}\right\}$$

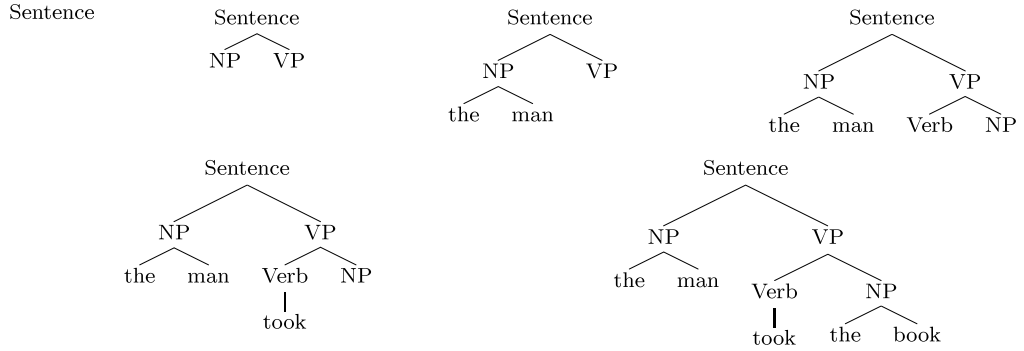
Chomsky's parse tree is based on the following CFG:

$$\begin{array}{l}\text{Sentence} \longrightarrow \text{NP VP}\\ \text{NP} \longrightarrow \text{the man}\\ \text{NP} \longrightarrow \text{the book}\\ \text{VP} \longrightarrow \text{Verb NP}\\ \text{Verb} \longrightarrow \text{took}\end{array}$$

[7] As a consequence, the terminal and non-terminal symbols of a given CFG form two finite and disjoint sets.

[8] Note, however, that so-called **free-word-order languages** (like Czech, German, or Russian) permit many different ways of ordering the words in a sentence (without a change in meaning). Instead of word order, these languages use case markings to indicate who did what to whom.

The CFG's terminal symbols are {the, man, took, book}, its non-terminal symbols are {Sentence, NP, VP, Verb}, and its starting symbol is "Sentence". Now, we are coming back to the beginning of the section, where we mentioned that a CFG is usually thought of in two ways: as a generator **or** as a parser. As a generator, the example CFG might produce the following series of intermediate parse trees (only the last one will be submitted to the generator's output):

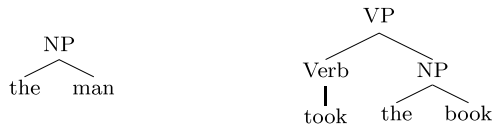


Starting with the starting symbol, each of these intermediate parse trees is generated by applying one rule of the CFG to a suitable non-terminal leaf node of the previous parse tree, thereby adding the CFG rule as a local tree. The generator stops, if all leaf nodes of the current parse tree are terminal nodes. The whole generation process, of course, is non-deterministic, and this fact will lead us later on directly to probabilistic CFGs. As a parser, instead, the example CFG has to deal with an input sentence like

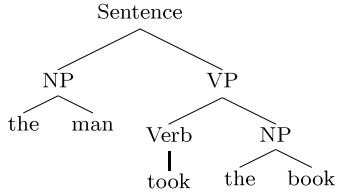
"the man took the book"

Usually, the parser starts processing the input sentence by assigning the words some local trees:

Then, the parser tries to add more local trees, by processing all the non-terminal nodes found in previous steps:



Doing this recursively, the parser provides us with a parse tree of the input sentence:



The example CFG is unambiguous for the given input sentence. Note, however, that this is far away from being the common situation. Usually, the parser stops, if all parse trees of the input sentence have been generated (and submitted to the output).

Now, we demonstrate that the fact that we can understand CFGs as generators leads directly to the **probabilistic context-free grammar** or **PCFG** formalism. As we already demonstrated for the generation process, the rules of the CFG serve as local trees that are incrementally used to build up a full parse tree (i.e. a parse tree without any non-terminal leaf nodes). This process, however, is non-deterministic: At most of its steps, some sort of **random choice** is involved that selects one of the different CFG rules which can potentially be appended to one of the non-terminal leaf nodes of the current parse tree[9]. Here is an example in the context of the generation process displayed above. For the CFG underlying Chomsky's very first parse tree, the non-terminal symbol NP is the left-hand side of two rules:

$$\begin{aligned} \text{NP} &\longrightarrow \text{the man} \\ \text{NP} &\longrightarrow \text{the book} \end{aligned}$$

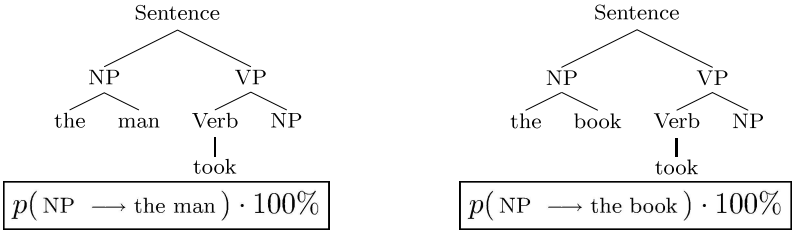
Clearly, when using the underlying CFG as a generator, we have to select either the first or the second rule, whenever a local NP tree shall be appended to the partial-parse tree given in the actual generation step. The choice might be either **fair** (both rules are chosen with probability 0.5) or **unfair** (the first rule is chosen, for example, with probability 0.9 and the second one with probability 0.1). In either case, a random choice between competing rules can be described by probability values which are directly allocated to the rules:

$$0 \leq p(\,\text{NP} \longrightarrow \text{the man}\,) \leq 1 \quad \text{and} \quad 0 \leq p(\,\text{NP} \longrightarrow \text{the book}\,) \leq 1$$

such that

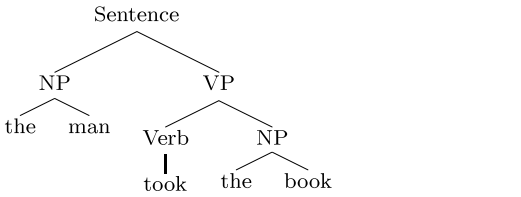
$$p(\,\text{NP} \longrightarrow \text{the man}\,) \;+\; p(\,\text{NP} \longrightarrow \text{the book}\,) = 1$$

Now, having these probabilities at hand, it turns out that it is even possible to predict how often the generator will produce the one or the other of the following alternate partial-parse trees:

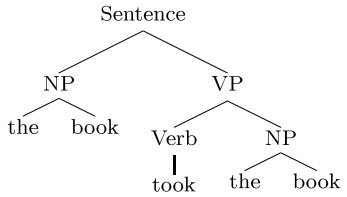


In turn, having this result at hand, we can also predict how often the generator will produce full-parse trees, for example, Chomsky's very first parse tree, or the parse tree of the sentence "the book took the book":

[9]Clearly, the final output of the generator is directly affected by the specific rule that has been selected by this random choice. Note also that there is another type of uncertainty in the generation process, playing, however, only a minor role: the specific place at which a CFG rule is to be appended does obviously not affect the generator's final output. So, these places can be deterministically chosen. For the generation process displayed above, for example, we decided to append the local trees always to the left-most non-terminal node of the actual partial-parse tree.



$p(\text{NP} \longrightarrow \text{the man}) \cdot p(\text{NP} \longrightarrow \text{the book}) \cdot 100\%$



$p(\text{NP} \longrightarrow \text{the book}) \cdot p(\text{NP} \longrightarrow \text{the book}) \cdot 100\%$

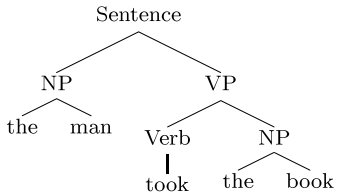
So, if $p(\text{NP} \longrightarrow \text{the man}) = 0.9$ and $p(\text{NP} \longrightarrow \text{the book}) = 0.1$, then it is nine times more likely that the generator produces Chomsky's very first parse tree. In the following, we are trying to generalize this result even a bit more. As we saw, there are three rules in the CFG, which cause no problems in terms of uncertainty. These are:

$$\begin{aligned} \text{Sentence} &\longrightarrow \text{NP VP} \\ \text{VP} &\longrightarrow \text{Verb NP} \\ \text{Verb} &\longrightarrow \text{took} \end{aligned}$$

To be more specific, we saw that these three rules have been always deterministically added to the partial-parse trees of the generation process. In terms of probability theory, determinism is expressed by the fact that certain events occur with a probability of one. In other words, a generator selects each of these rules with a probability of 100%, either when starting the generation process, or when expanding a VP or a Verb non-terminal node. So, we let

$$\begin{aligned} &p(\text{Sentence} \longrightarrow \text{NP VP}) = 1 \\ &p(\text{VP} \longrightarrow \text{Verb NP}) = 1 \\ &p(\text{Verb} \longrightarrow \text{took}) = 1 \end{aligned}$$

The question is now: Have we won something by treating also the deterministic choices as probabilistic events? The answer is yes: A closer look at our example reveals that we can now predict easily how often the generator will produce a specific parse tree: The likelihood of a CFG's parse tree can be simply calculated as the product of the probabilities of all rules occurring in the tree. For example:



$p(\text{S} \longrightarrow \text{NP VP}) \cdot p(\text{NP} \longrightarrow \text{the man}) \cdot p(\text{VP} \longrightarrow \text{Verb NP}) \cdot p(\text{Verb} \longrightarrow \text{took}) \cdot p(\text{NP} \longrightarrow \text{the book})$

To wrap up, we investigated the small CFG underlying Chomsky's very first parse tree. Motivated by the fact that a CFG can be used as a generator, we assigned each of its rules a weight (a non-negative real number) such that the weights of all rules with the same left-hand side sum up to one. In other words, all CFG fragments (comprising the CFG rules with the same left-hand side) have been assigned a probability distribution, as displayed in the following table:

CFG rule	**Rule probability**	
Sentence $\longrightarrow$ NP VP	$p(\text{Sentence} \longrightarrow \text{NP VP}) = 1$	
NP $\longrightarrow$ the man	$p(\text{NP} \longrightarrow \text{the man})$	} **summing to 1**
NP $\longrightarrow$ the book	$p(\text{NP} \longrightarrow \text{the book})$	}
VP $\longrightarrow$ Verb NP	$p(\text{VP} \longrightarrow \text{Verb NP}) = 1$	
Verb $\longrightarrow$ took	$p(\text{Verb} \longrightarrow \text{took}) = 1$	

As a result, the likelihood of each of the grammar's parse trees (when using the CFG as a generator) can be calculated by multiplying the probabilities of all rules occurring in the tree. This observation leads directly to the standard definition of a probabilistic context-free grammar, as well as to the definition of probabilities for parse-trees.

Definition 16 *A pair $< G, p >$ consisting of a context-free grammar G and a probability distribution $p\colon \mathcal{X} \to [0,1]$ on the set $\mathcal{X}$ of all finite full-parse trees of G is called a* ***probabilistic context-free grammar*** *or* ***PCFG****, if for all parse trees $x \in \mathcal{X}$*

$$p(x) = \prod_{r \in G} p(r)^{f_r(x)}$$

Here, $f_r(x)$ is the number of occurrences of the rule r in the tree x, and $p(r)$ is a probability allocated to the rule r, such that for all non-terminal symbols A

$$\sum_{r \in G_A} p(r) = 1$$

where $G_A = \{\, r \in G \mid lhs(r) = A\}$ is the grammar fragment comprising all rules with the left-hand side A. In other words, a probabilistic context-free grammar is defined by a context-free grammar G and some probability distributions on the grammar fragments G_A, thereby inducing a probability distribution on the set of all full-parse trees.

So far, we have not checked for our example that the probabilities of all full-parse trees are summing up to one. According to Definition 16, however, this is **the fundamental property** of PCFGs (and it should be really checked for every PCFG which is accidentally given to us). Obviously, the example grammar has four full-parse trees, and the sum of their probabilities can be calculated as follows (by omitting all rules with a probability of one):

$$\begin{aligned}
p(\mathcal{X}) &= p(\text{NP} \longrightarrow \text{the man}) \cdot p(\text{NP} \longrightarrow \text{the book}) + \\
&\quad p(\text{NP} \longrightarrow \text{the book}) \cdot p(\text{NP} \longrightarrow \text{the book}) + \\
&\quad p(\text{NP} \longrightarrow \text{the man}) \cdot p(\text{NP} \longrightarrow \text{the man}) + \\
&\quad p(\text{NP} \longrightarrow \text{the book}) \cdot p(\text{NP} \longrightarrow \text{the man}) \\
&= \Big(p(\text{NP} \longrightarrow \text{the man}) + p(\text{NP} \longrightarrow \text{the book})\Big) \cdot p(\text{NP} \longrightarrow \text{the book}) + \\
&\quad \Big(p(\text{NP} \longrightarrow \text{the man}) + p(\text{NP} \longrightarrow \text{the book})\Big) \cdot p(\text{NP} \longrightarrow \text{the man}) \\
&= 1
\end{aligned}$$

For the last equation, we are using three times that p is a probability distribution on the grammar fragment G_{NP}, i.e., we are exploiting that $p(\text{NP} \longrightarrow \text{the man}) + p(\text{NP} \longrightarrow \text{the book}) = 1$.

The following examples show that we really have to do this kind of "probabilistic grammar checking". We are presenting two non-standard PCFGs: The first one consists of the rules

$$\begin{array}{lll} \mathrm{S} & \longrightarrow \mathrm{NP\ sleeps} & (1.0) \\ \mathrm{S} & \longrightarrow \mathrm{John\ sleeps} & (0.7) \\ \mathrm{NP} & \longrightarrow \mathrm{John} & (0.3) \end{array}$$

The second one is a well-known highly-recursive grammar (Chi and Geman 1998), and it is given by

$$\begin{array}{lll} \mathrm{S} & \longrightarrow \mathrm{S\ S} & (\mathrm{q}) \\ \mathrm{S} & \longrightarrow \mathrm{a} & (\mathrm{1\text{-}q}) \end{array} \quad \text{with } 0.5 < q \leq 1$$

What is wrong with these grammars? Well, the first grammar provides us with a probability distribution on its full-parse trees, as can be seen here

On each of its grammar fragments, however, the rule probabilities do not form a probability distribution (neither on G_{S} nor on G_{NP}). The second grammar is even worse: We do have a probability distribution on G_{S}, but even so, we do not have a probability distribution on the set of full-parse trees (because their probabilities are summing to less than one[10]).

[10] This can be proven as follows: Let $\mathcal{T}$ be the set of all finite full-parse trees that can be generated by the given context-free grammar. Then, it is easy to verify that $\pi := p(\mathcal{T})$ is a solution of the following equation

$$\pi = 1 - q + q \cdot \pi \cdot \pi$$

Here, $1 - q$ is the probability of the tree

, whereas $q \cdot \pi \cdot \pi$ corresponds to the forest 

.

It is easy to check that the derived quadratic equation has two solutions: $\pi_1 = 1$ and $\pi_2 = \frac{1-q}{q}$. Note that it is quite natural that **two** solutions arise: The set of all "infinite full-parse trees" matches also our under-specified approach of calculating π. Now, in the case of $0.5 < q \leq 1$, it turns out that the set of infinite trees is allocated a proper probability $\pi_1 = 1$. (For the special case $q = 1$, this can be intuitively verified: The generator will never touch the rule $S \longrightarrow a$, and therefore, this special PCFG produces infinite parse trees only.) As a consequence, all finite full-parse trees is allocated the total probability π_2. In other words, $p(\mathcal{T}) = \frac{1-q}{q} < 1$. In a certain sense, however, we are able to repair both grammars. For example,

$$\begin{array}{lll} \mathrm{S} & \longrightarrow \mathrm{NP\ sleeps} & (0.3) \\ \mathrm{S} & \longrightarrow \mathrm{John\ sleeps} & (0.7) \\ \mathrm{NP} & \longrightarrow \mathrm{John} & (1.0) \end{array}$$

is the standard-PCFG counterpart of the first grammar, where

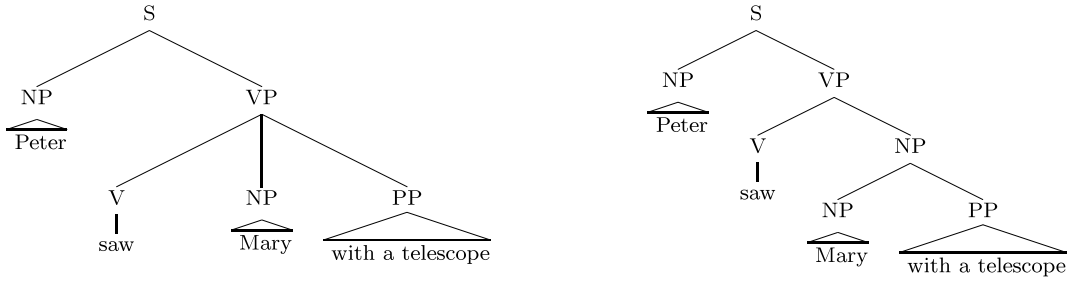
$$\begin{array}{lll} \mathrm{S} & \longrightarrow \mathrm{S\ S} & (\mathrm{1\text{-}q}) \\ \mathrm{S} & \longrightarrow \mathrm{a} & (\mathrm{q}) \end{array} \quad \text{with } 0.5 < q \leq 1$$

is a standard-PCFG counterpart of the second grammar: The first grammar and its counterpart provide us with exactly the same parse-tree probabilities, while the second grammar and its counterpart produce parse-tree probabilities, which are proportional to each other. Especially for the second example, this interesting result is a special case of an important general theorem recently proven by Nederhof and Satta (2003). Sloppily formulated, their Theorem 7 states that: *For each **weighted CFG** (defined on the basis of **rule weights** instead of rule probabilities) is a standard PCFG with the same symbolic backbone, such that (i) the parse-tree probabilities (produced by the PCFG) are summing to one, and (ii) the parse-tree weights (produced by the weighted CFG) are proportional to the parse-tree probabilities (produced by the PCFG).*
As a consequence, we are getting what we really want: Applied to ambiguity resolution, the original grammars and their counterparts provide us with exactly the same maximum-probability-parse trees.

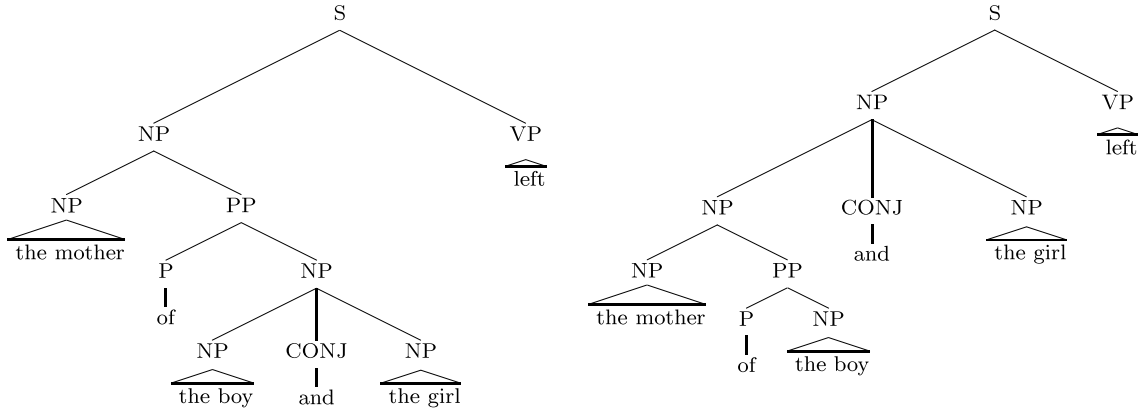
Background: Resolving Ambiguities with PCFGs

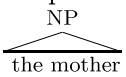
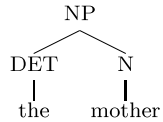
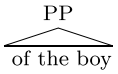
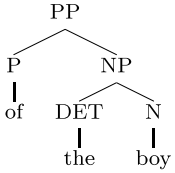
A property of most formalizations of natural language in terms of CFGs is **ambiguity**: the fact that sentences have more than one possible phrase structure (and therefore more than one meaning). Here are two prominent types of ambiguity:

Ambiguity caused by prepositional-phrase attachment:



Ambiguity caused by conjunctions:



As usual in computational linguistics, some phrase structures have been displayed in abbreviated form: For example, the term  is used as a short form for the parse tree  , and the term  is a place holder for the even more complex parse tree  .

In both examples, the ambiguity is caused by the fact that the underlying CFG contains **recursive rules**, i.e., rules that can be applied an arbitrary number of times. Clearly, the rules NP $\longrightarrow$ NP CONJ NP and NP $\longrightarrow$ NP PP belong to this type, since they can be used to generate nominal phrases of an arbitrary length. The rules VP $\longrightarrow$ V NP and

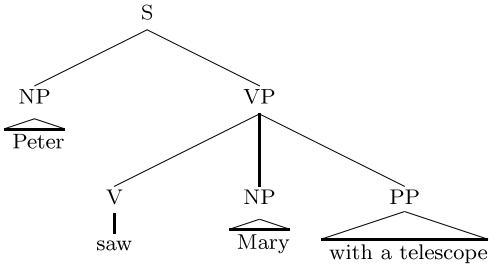
PP $\longrightarrow$ P NP, however, might be also called (indirectly) recursive, since they can generate verbal and prepositional phrases of an arbitrary length (in combination with NP $\longrightarrow$ NP PP). Besides ambiguity, recursivity makes it also possible that two words that are generated by the same CFG rule (i.e. which are syntactically linked) can occur far apart in a sentence:

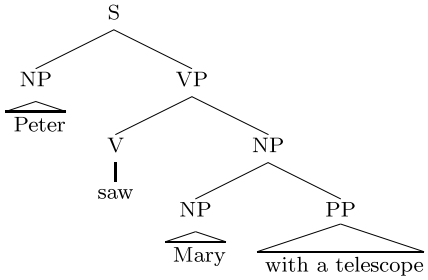
The **bird** with the nice brown eyes and the beautiful tail feathers **catches** a worm.

These types of phenomena are called **non-local dependencies**, and it is important to note that non-local phenomena (which can be handled by CFGs) are beyond the scope of many popular models that focus on modeling local dependencies (such as n-gram, Markov, and hidden Markov models[11]). So, a part-of-speech tagger (based on a HMM model) might have difficulties with sentences like the one we mentioned, because it will not expect that a singular verb occurs after a plural noun.

Having this at hand, of course, the central question is now: Can PCFGs handle ambiguity? The somewhat surprising answer is: Yes, but the symbolic backbone of the PCFG plays a major role in solving this difficult task. To be a bit more specific, the CFG underlying the given PCFG has to have some good properties, or the other way round, probabilistic modeling of some "weak" CFGs may result in PCFGs which can not resolve the CFG's ambiguities. From a probabilistic modeler's point of view, there is really some non-trivial relation between such tasks as "writing a formal grammar" and "modeling a probabilistic grammar". So, we are convinced that formal-grammar writers should help probabilistic-grammar modelers, and the other way round.

To exemplify this, we will have a closer look at the examples above, where we presented two common types of ambiguity. In general, a PCFG resolves ambiguity (i) by calculating all the full parse-trees of a given sentence (using the symbolic backbone of the CFG), and (ii) by allocating probabilities to all these trees (using the rule probabilities of the PCFG), and finally (iii) by choosing the most probable parse as the analysis of the given sentence. According to this procedure, we are calculating, for example





$$p(\text{S} \longrightarrow \text{NP VP}) \cdot p\left(\begin{array}{c}\text{NP}\\ \triangle\\ \text{Peter}\end{array}\right) \cdot p(\mathbf{VP} \longrightarrow \mathbf{V\ NP\ PP}) \cdot p(\text{V} \longrightarrow \text{saw}) \cdot p\left(\begin{array}{c}\text{NP}\\ \triangle\\ \text{Mary}\end{array}\right) \cdot p\left(\begin{array}{c}\text{PP}\\ \triangle\\ \text{with a telescope}\end{array}\right)$$

$$p(\text{S} \longrightarrow \text{NP VP}) \cdot p\left(\begin{array}{c}\text{NP}\\ \triangle\\ \text{Peter}\end{array}\right) \cdot p(\mathbf{VP} \longrightarrow \mathbf{V\ NP}) \cdot p(\mathbf{NP} \longrightarrow \mathbf{NP\ PP}) \cdot p(\text{V} \longrightarrow \text{saw}) \cdot p\left(\begin{array}{c}\text{NP}\\ \triangle\\ \text{Mary}\end{array}\right) \cdot p\left(\begin{array}{c}\text{PP}\\ \triangle\\ \text{with a telescope}\end{array}\right)$$

[11] It is well-known, however, that a CFG without **center-embeddings** can be transformed to a regular grammar (the symbolic backbone of a hidden Markov model).

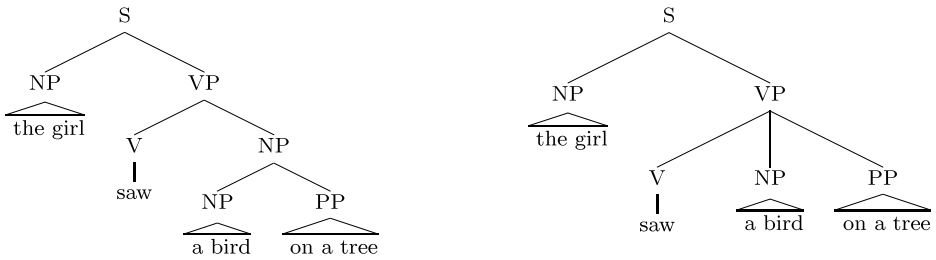
Comparing the probabilities for these two analyzes, we are choosing the analysis at the left-hand side of this figure, if

$$p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP\ PP}\,) > p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP}\,) \cdot p(\,\mathbf{NP}\ \longrightarrow \mathbf{NP\ PP}\,)$$

So, in principle, the PP-attachment ambiguity encoded in this CFG **can** be solved by a probabilistic model built on top of this CFG. Moreover, it is especially nice that such a PCFG resolves the ambiguity by looking only at those rules of the CFG, which directly cause the PP-attachment ambiguity.

So far, so good: We are able to use PCFGs in order to select between competing analyzes of a sentence. Looking at the second example (ambiguity caused by conjunctions), however, we are faced with a serious problem: Both trees have a different structure, but exactly the same context-free rules are used for generating these different structures. As a consequence, both trees are allocated the same probability (independently from the specific rule probabilities which might have been offered to you by the very best estimation methods). So, any PCFG based on the given CFG is unable to resolve the ambiguity manifested in the two trees.

Here is another problem. Using the grammar underlying our first example, the sentence "the girl saw a bird on a tree" has the following two analyzes



Comparing the probabilities for these two analyzes, we are choosing the analysis at the left-hand side of this figure, if

$$p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP}\,) \cdot p(\,\mathbf{NP}\ \longrightarrow \mathbf{NP\ PP}\,) > p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP\ PP}\,)$$

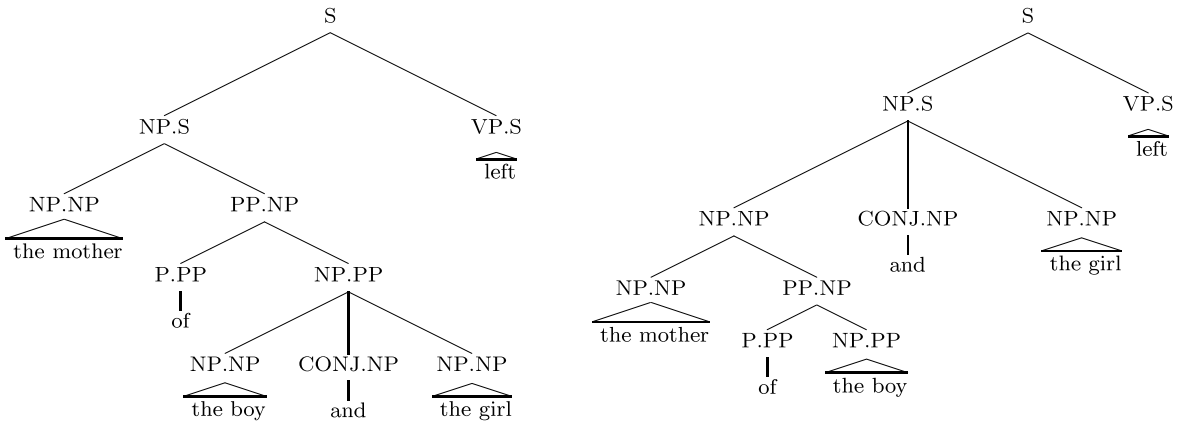
Relating this result to the disambiguation result of the sentence "John saw Mary with a telescope", the prepositional phrases are attached in both cases either to the verbal phrase or to the nominal phrase. Instead, it seems to be more plausible that the PP "with the telescope" is attached to the verbal phrase, whereas the PP "on a tree" is attached to the noun phrase.

Obviously, there is only one possible solution to this problem: We have to re-write the given CFG in such a way that a probabilistic model will be able to assign different probabilities to different analyzes. For our last example, it is sufficient to enrich the underlying CFG with some simple PP markup, enabling in principle that

$$\begin{aligned} p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP\ PP\text{-}ON}\,) &< p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP}\,) \cdot p(\,\mathbf{NP}\ \longrightarrow \mathbf{NP\ PP\text{-}ON}\,) \\ p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP\ PP\text{-}WITH}\,) &> p(\,\mathbf{VP}\ \longrightarrow \mathbf{V\ NP}\,) \cdot p(\,\mathbf{NP}\ \longrightarrow \mathbf{NP\ PP\text{-}WITH}\,) \end{aligned}$$

Of course, other linguistically more sophisticated modifications of the original CFG (that handle e.g. agreement information, sub-cat frames, selectional preferences, etc) are also welcome. Our only request is that the modified CFGs lead to PCFGs which are able to resolve

the different types of ambiguities encoded in the original CFG. Now, writing and re-writing a formal grammar is a job that grammar writers can do probably much better than modelers of probabilistic grammars. In the past, however, writers of formal grammars seemed to be uninterested in this specific task, or they are still unaware of its existence. So, modelers of PCFGs regard it nowadays also as an important part of their job to transform a given CFG in such a way that probabilistic versions of the modified CFG are able to resolve ambiguities of the original CFG. During the last years, a bunch of automatic grammar-transformation techniques have been developed, which offer some interesting solutions to this quite complex problem. Where the work of Klein and Manning (2003) describes one of the latest approaches to semi-automatic grammar-transformation, the **parent-encoding technique** introduced by Johnson (1998) is the earliest and purely automatic grammar-transformation technique: For each local tree, the parent's category is appended to all daughter categories. Using the example above, where we showed that conjunctions cause ambiguities, the parent-encoded trees are looking as follows:



Clearly, parent-encoding of the original trees may result in different probabilities of the transformed trees: In this example, we will choose the analysis at the left-hand side, if

$$p(\,\mathbf{NP.S}\ \longrightarrow \text{NP.NP PP.NP}\,)\cdot p(\,\mathbf{NP.PP}\ \longrightarrow \text{NP.NP CONJ.NP NP.NP}\,)\cdot p\left(\begin{array}{c}\mathbf{NP.NP}\\ \triangle\\ \text{the boy}\end{array}\right)$$

is more likely than

$$p(\,\mathbf{NP.NP}\ \longrightarrow \text{NP.NP PP.NP}\,)\cdot p(\,\mathbf{NP.S}\ \longrightarrow \text{NP.NP CONJ.NP NP..NP}\,)\cdot p\left(\begin{array}{c}\mathbf{NP.PP}\\ \triangle\\ \text{the boy}\end{array}\right)$$

As in the example before, it is again nice to see that these probabilities are pin-pointing exactly at those rules of the underlying grammar which have introduced the ambiguity.

In the rest of the section, we will present the notion of treebank grammars, which can be informally described as PCFGs that are constructed on the basis of a corpus of full-parse trees (Charniak 1996). We will demonstrate that treebank grammars can resolve the ambiguous sentences of the treebank (as well as ambiguous similar sentences), if the treebank mark-up is rich enough to distinguish between the different types of ambiguities that are encoded in the treebank.

Definition 17 *For a given **treebank**, i.e., for a non-empty and finite corpus of full-parse trees, the **treebank grammar** $< G, p >$ is a PCFG defined by*

(i) G is the context-free grammar read off from the treebank, and

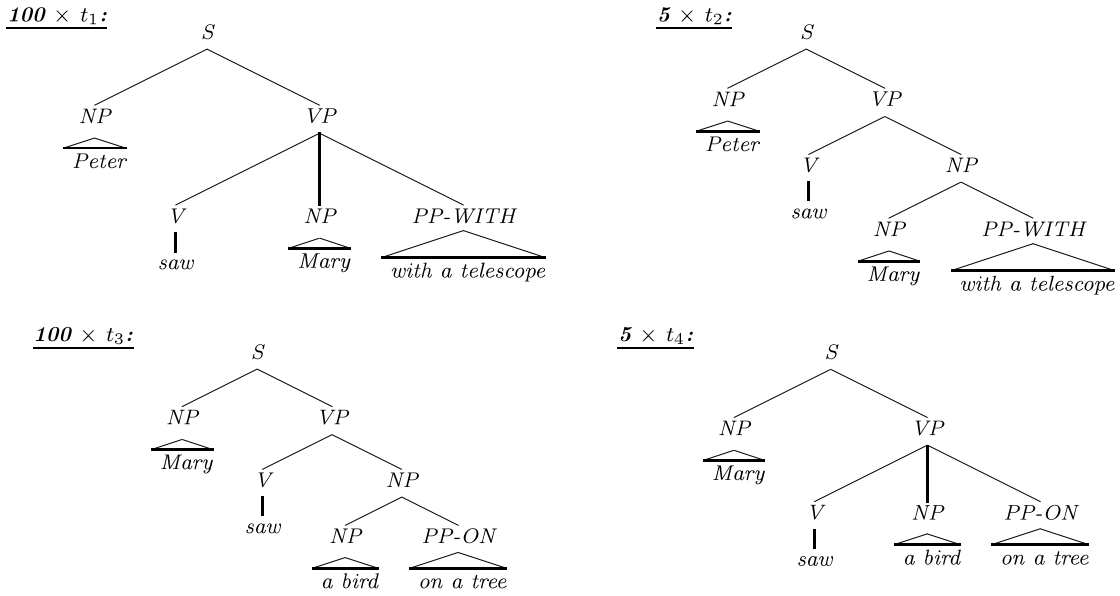
(ii) p is the probability distribution on the set of full-parse trees of G, induced by the following specific probability distributions on the grammar fragments G_A:

$$p(r) = \frac{f(r)}{\sum_{r \in G_A} f(r)} \qquad \text{for all } r \in G_A$$

Here, $f(r)$ is the number of times a rule $r \in G$ occurs in the treebank.

Note: Later on (see Theorem 10), we will show that each treebank grammar is the unique maximum-likelihood estimate of G's probability model on the given treebank. So, it is especially guaranteed that p is a probability distribution on the set of full-parse trees of G, and that $< G, P >$ is a standard PCFG (see Definition 16).

Example 6 *We shall now consider a treebank given by the following 210 full-parse trees:*



We shall (i) generate the treebank grammar and (ii) using this treebank grammar, we shall resolve the ambiguities of the sentences occurring in the treebank.

Ad (i): The following table displays the rules r of the CFG encoded in the given treebank (thereby assuming for the sake of simplicity that the NP and PP non-terminals expand directly to terminal symbols), the rule frequencies $f(r)$ i.e. the number of times a rule occurs in the treebank, as well as the rule probabilities $p(r)$ as defined in Definition 17.

CFG rule	Rule frequency	Rule probability
S $\longrightarrow$ NP VP	$100 + 5 + 100 + 5$	$\frac{210}{210} = 1.000$
VP $\longrightarrow$ V NP PP-WITH	100	$\frac{100}{210} \approx 0.476$
VP $\longrightarrow$ V NP PP-ON	5	$\frac{5}{210} \approx 0.024$
VP $\longrightarrow$ V NP	$5 + 100$	$\frac{105}{210} = 0.500$
NP $\longrightarrow$ Peter	$100 + 5$	$\frac{105}{525} = 0.200$
NP $\longrightarrow$ Mary	$100 + 5 + 100 + 5$	$\frac{210}{525} = 0.400$
NP $\longrightarrow$ a bird	$100 + 5$	$\frac{105}{525} = 0.200$
NP $\longrightarrow$ NP PP-WITH	5	$\frac{5}{525} \approx 0.010$
NP $\longrightarrow$ NP PP-ON	100	$\frac{100}{525} \approx 0.190$
PP-WITH $\longrightarrow$ with a telescope	$100 + 5$	$\frac{105}{105} = 1.000$
PP-ON $\longrightarrow$ on a tree	$100 + 5$	$\frac{105}{105} = 1.000$
V $\longrightarrow$ saw	$100 + 5 + 100 + 5$	$\frac{210}{210} = 1.000$

Ad (ii): As we have already seen, the treebank grammar selects the full-parse tree t_1 of the sentence "Peter saw Mary with a telescope", if

$$p(\,\textbf{VP} \;\longrightarrow \textbf{V NP PP-WITH}\,) > p(\,\textbf{VP} \;\longrightarrow \textbf{V NP}\,) \cdot p(\,\textbf{NP} \;\longrightarrow \textbf{NP PP-WITH}\,)$$

Using the approximate probabilities for these rules, this is indeed true: $0.476 > 0.500 \cdot 0.010$. (Note that exactly the same argument can be applied to similar but more complex sentences like "Peter saw a bird on a tree with a telescope".) For the second sentence occurring in the treebank, "Mary saw a bird on a tree", the treebank grammar selects the full-parse tree t_3, if

$$p(\,\textbf{VP} \;\longrightarrow \textbf{V NP PP-ON}\,) < p(\,\textbf{VP} \;\longrightarrow \textbf{V NP}\,) \cdot p(\,\textbf{NP} \;\longrightarrow \textbf{NP PP-ON}\,)$$

Indeed, this is the case: $0.024 < 0.500 \cdot 0.190$.

Maximum-Likelihood Estimation of PCFGs

So far, we have seen that probabilistic context-free grammars can be used to resolve the ambiguities that are caused by their underlying context-free backbone, and we noted already that certain grammar-transformation are sometimes necessary to achieve this goal. All these application features are displayed in a "horizontal view" in Figure 7. In what follows next, we will concentrate on the "vertical view" of this figure. To be more specific, we will focus on the following two questions.

(i) how to characterize the probability model of a given context-free grammar, and

(ii) second, how to estimate an appropriate instance of the context-free grammar's probability model, if a corpus of input data is additionally given.

The latter question is a tough one: It is true that the treebank-training method, which we defined in the previous section more or less heuristically, leads to PCFGs that are able to

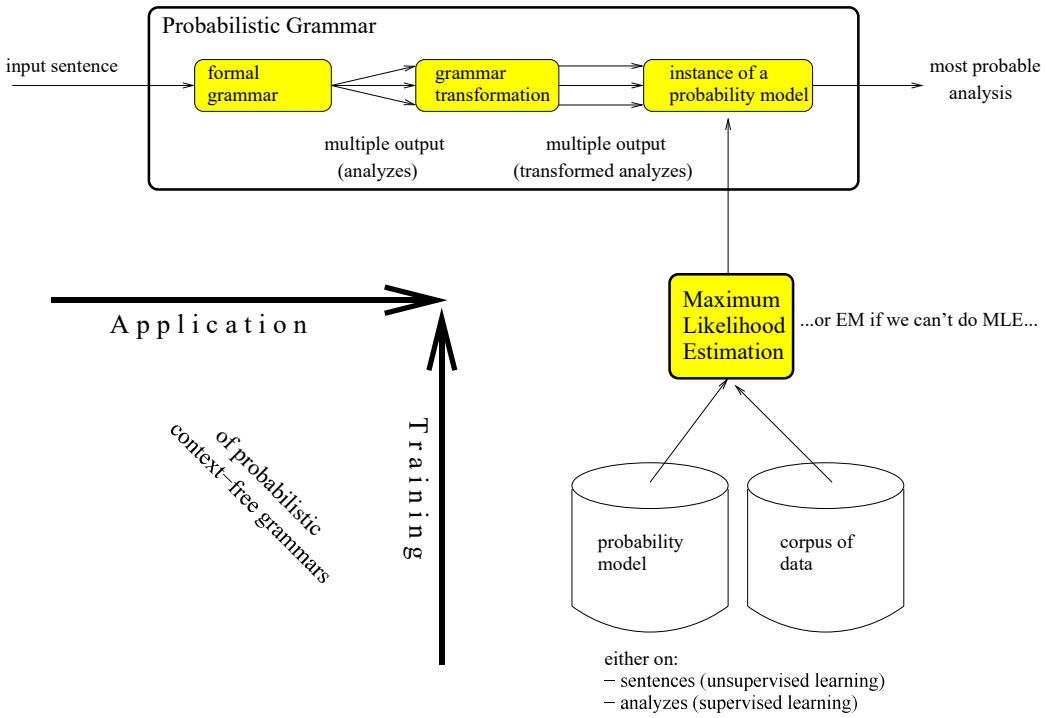


Figure 7: Application and training of probabilistic context-free grammars

resolve ambiguities. From what we have done so far in this section, however, we have no clear idea how the treebank-training method is related to maximum-likelihood estimation or the EM training method. So, let us start with the first question.

Definition 18 *Let G be a context-free grammar, and let $\mathcal{X}$ be the set of full-parse trees of G. Then, the* ***probability model of*** *G is defined by*

$$\mathcal{M}_G = \left\{ p \in \mathcal{M}(\mathcal{X}) \;\middle|\; p(x) = \prod_{r \in G} p(r)^{f_r(x)} \text{ with } \sum_{r \in G_A} p(r) = 1 \text{ for all grammar fragments } G_A \right\}$$

In other words, each instance p of the probability model $\mathcal{M}_G$ is associated to a probabilistic context-free grammar $< G, p >$ having the context-free backbone G. (See Definition 16 for the meaning of the terms $p(r)$, $f_r(x)$ and G_A.)

As we have already seen, there are some non-standard PCFGs (like $\text{S} \longrightarrow \text{S S}\ (0.9), \text{S} \longrightarrow \text{a}\ (0.1)$) which do not induce a probability distribution on the set of full-parse trees. This gives us a rough idea that it might be quite difficult to characterize those PCFGs $< G, p >$ which are associated to an instance p of the unrestricted probability model $\mathcal{M}(\mathcal{X})$. In other words, it might be quite difficult to characterize G's probability model $\mathcal{M}_G$. For calculating a maximum-likelihood estimate of $\mathcal{M}_G$ on a corpus $f_{\mathcal{T}}$ of full-parse trees, however, we have to solve this task. For example, if we are targeting to exploit the powerful Theorem 1, we have to prove either that $\mathcal{M}_G$ equals the unrestricted probability model $\mathcal{M}(\mathcal{X})$, or that the relative-frequency estimate $\tilde{p}$ on $f_{\mathcal{T}}$ is an instance of $\mathcal{M}_G$. For most context-free grammars

G, however, the following theorems show that this is a too simplistic approach of finding a maximum-likelihood estimate of $\mathcal{M}_G$.

Theorem 8 *Let G be a context-free grammar, and let $\tilde{p}$ be the relative-frequency estimate on a non-empty and finite corpus $f_{\mathcal{T}} \colon \mathcal{X} \to \mathcal{R}$ of full-parse trees of G. Then $\tilde{p} \notin \mathcal{M}_G$ if*

(i) G can be read off from $f_{\mathcal{T}}$, and

(ii) G has a full-parse tree $x_\infty \in \mathcal{X}$ that is not in $f_{\mathcal{T}}$.

Proof Assume that $\tilde{p} \in \mathcal{M}_G$. In what follows, we will show that this assumption leads to a contradiction. First, by definition of $\mathcal{M}_G$, it follows that there are some weights $0 \leq \pi(r) \leq 1$ such that

$$\tilde{p}(x) = \prod_{r \in G} \pi(r)^{f_r(x)} \qquad \text{for all } x \in \mathcal{X}$$

We will show next that $\pi(r) > 0$ for all $r \in G$.

> Assume that there is a rule $r_0 \in G$ with $\pi(r_0) = 0$. By (i), G can be read off from $f_{\mathcal{T}}$. So, $f_{\mathcal{T}}$ contains a full-parse tree x_0 such that the rule r_0 occurs in x_0, i.e.,
>
> $$f_{\mathcal{T}}(x_0) > 0 \quad \text{and} \quad f_{r_0}(x_0) > 0$$
>
> It follows both $\tilde{p}(x_0) = \frac{f_{\mathcal{T}}(x_0)}{|f_{\mathcal{T}}|} > 0$ and $\tilde{p}(x_0) = \prod_{r \in G} \pi(r)^{f_r(x_0)} = \cdots \pi(r_0)^{f_{r_0}(x_0)} \cdots = 0$, which is a contradiction.

Therefore

$$\tilde{p}(x) = \prod_{r \in G} \pi(r)^{f_r(x)} > 0 \qquad \text{for all } x \in \mathcal{X}$$

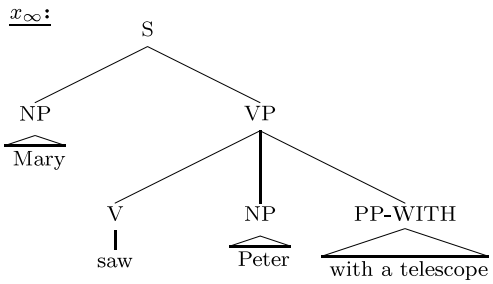
On the other hand by (ii), there is the full-parse tree $x_\infty \in \mathcal{X}$ which is not in $f_{\mathcal{T}}$. So, $\tilde{p}(x_\infty) = \frac{f_{\mathcal{T}}(x_\infty)}{|f_{\mathcal{T}}|} = 0$, which is a contradiction to the last inequality **q.e.d.**

Example 7 *The treebank grammar described in Example 6 is a context-free grammar of the type described in Theorem 8.*

The relative-frequency estimate $\tilde{p}$ on the treebank is given by:

$$\tilde{p}(t_1) = \tilde{p}(t_3) = \frac{100}{210} \qquad \text{and} \qquad \tilde{p}(t_2) = \tilde{p}(t_4) = \frac{5}{210}$$

All other full-parse trees of the treebank grammar get allocated a probability of zero by the relative-frequency estimate. So, for example, $\tilde{p}(x_\infty) = 0$ for



As a consequence, $\tilde{p}$ can not be an instance of the probability model of the treebank grammar: Otherwise, $\tilde{p}$ would allocate both full-parse trees t_1 and x_∞ exactly the same probabilities (because t_1 and x_∞ contain exactly the same rules).

Theorem 9 *For each context-free grammar G with an infinite set $\mathcal{X}$ of full-parse trees, the probability model of G is restricted*

$$\mathcal{M}_G \neq \mathcal{M}(\mathcal{X})$$

Proof First, each context-free grammar consists of a finite number of rules. Thus it is possible to construct a treebank, such that G is encoded by the treebank. (Without loss of generality, we are assuming here that all non-terminal symbols of G are reachable and productive.) So, let $f_\mathcal{T}$ be a non-empty and finite corpus of full-parse trees of G such that G can be read off from $f_\mathcal{T}$, and let $\tilde{p}$ be the relative-frequency estimate on $f_\mathcal{T}$. Second, using $|\mathcal{X}| = \infty$ and $|f_\mathcal{T}| < \infty$, it follows that there is at least one full-parse tree $x_\infty \in \mathcal{X}$ which is not in $f_\mathcal{T}$. So, the conditions of Theorem 8 are met, and we are concluding that $\tilde{p} \notin \mathcal{M}_G$. On the other hand, $\tilde{p} \in \mathcal{M}(\mathcal{X})$. So, clearly, $\mathcal{M}_G \neq \mathcal{M}(\mathcal{X})$. In other words, $\mathcal{M}_G$ is a restricted probability model **q.e.d.**

After all, we might recognize that the previous results are not bad. Yes, the probability model of a given CFG is restricted in most of the cases. The missing distributions, however, are the relative-frequency estimates on each treebank encoding the given CFG. These relative-frequency estimates lack the ability of any generalization power: They allocate each full-parse tree not being in the treebank a zero-probability. Obviously, however, we surely want a probability model that can be learned by maximum-likelihood estimation on a corpus of full-parse trees, but that is at the same time able to deal with full-parse trees not seen in the treebank. The following theorem shows that we have already found one.

Theorem 10 *Let $f_\mathcal{T} : \mathcal{X} \rightarrow \mathcal{R}$ be a non-empty and finite corpus of full-parse trees, and let $< G, p_\mathcal{T} >$ be the treebank grammar read off from $f_\mathcal{T}$. Then, $p_\mathcal{T}$ is a maximum-likelihood estimate of $\mathcal{M}_G$ on $f_\mathcal{T}$, i.e.,*

$$L(f_\mathcal{T}; p_\mathcal{T}) = \max_{p \in \mathcal{M}_G} L(f_\mathcal{T}; p)$$

Moreover, maximum-likelihood estimation of $\mathcal{M}_G$ on $f_\mathcal{T}$ yields a unique estimate.

This theorem is well-known. The following proof, however, is especially simple and (to the best of my knowledge) was given first by Prescher (2002).

Proof First step: We will show that for all model instances $p \in \mathcal{M}_G$

$$L(f_\mathcal{T}; p) = L(f; p)$$

At the right-hand side of this equation, f refers to the corpus of rules that are read off from the treebank $f_\mathcal{T}$, i.e., $f(r)$ is the number of times a rule $r \in G$ occurs in the treebank; Similarly, p refers to the probabilities $p(r)$ of the rules $r \in G$. In contrast, at the left-hand side of the equation, p refers to the probabilities $p(x)$ of the full-parse trees $x \in \mathcal{X}$. The proof of the equation is relatively straight-forward:

$$
\begin{aligned}
L(f_{\mathcal{T}};p) &= \prod_{x\in\mathcal{X}} p(x)^{f_{\mathcal{T}}(x)} \\
&= \prod_{x\in\mathcal{X}} \left(\prod_{r\in G} p(r)^{f_r(x)} \right)^{f_{\mathcal{T}}(x)} \\
&= \prod_{x\in\mathcal{X}} \prod_{r\in G} p(r)^{f_{\mathcal{T}}(x)\cdot f_r(x)} \\
&= \prod_{r\in G} \prod_{x\in\mathcal{X}} p(r)^{f_{\mathcal{T}}(x)\cdot f_r(x)} \\
&= \prod_{r\in G} p(r)^{\sum_{x\in\mathcal{X}} f_{\mathcal{T}}(x)\cdot f_r(x)} \\
&= \prod_{r\in G} p(r)^{f(r)} \\
&= L(f;p)
\end{aligned}
$$

In the 6^{th} equation, we simply used that $f(r)$ (the number of times a specific rule occurs in the treebank) can be calculated by summing up all the $f_r(x)$ (the number of times this rule occurs in a specific full-parse tree $x \in \mathcal{X}$):

$$f(r) = \sum_{x\in\mathcal{X}} f_{\mathcal{T}}(x) \cdot f_r(x)$$

So, maximizing $L(f_{\mathcal{T}};p)$ is equivalent to maximizing $L(f;p)$. Unfortunately, Theorem 1 can not be applied to maximize the term $L(f;p)$, because the rule probabilities $p(r)$ do not form a probability distribution on the set of all grammar rules. They do form, however, probability distributions on the grammar fragments G_A. So, we have to refine our result a bit more.

Second step: We are showing here that

$$L(f;p) = \prod_A L(f_A;p)$$

Here, each f_A is a corpus of rules, read off from the given treebank, thereby filtering out all rules not having the left-hand side A. To be specific, we define

$$f_A(r) = \begin{cases} f(r) & \text{if} \quad r \in G_A \\ 0 & \text{else} \end{cases}$$

Again, the proof is easy:

$$
\begin{aligned}
L(f;p) &= \prod_{r\in G} p(r)^{f(r)} \\
&= \prod_A \prod_{r\in G_A} p(r)^{f(r)} \\
&= \prod_A \prod_{r\in G_A} p(r)^{f_A(r)} \\
&= \prod_A L(f_A;p)
\end{aligned}
$$

Third step: Combining the first and second step, we conclude that

$$L(f_{\mathcal{T}};p) = \prod_A L(f_A;p)$$

So, maximizing $L(f_{\mathcal{T}};p)$ is equivalent to maximizing $\prod_A L(f_A;p)$. Now, a product is maximized, if all its factors are maximized. So, in what follows, we are focusing on how to maximize the terms $L(f_A;p)$. First of all, the corpus f_A comprises only rules with the left-hand side A. So, the value of $L(f_A;p)$ depends only on the values $p(r)$ of the rules $r \in G_A$. These values, however, form a probability distribution on G_A, and all these probability distributions on G_A have to be considered for maximizing $L(f_A;p)$. It follows that we have to calculate an instance $\hat{p}_A$ of the unrestricted probability model $\mathcal{M}(G_A)$ such that

$$L(f_A;\hat{p}_A) = \max_{p \in \mathcal{M}(G_A)} L(f_A;p)$$

In other words, we have to calculate a maximum-likelihood estimate of the unrestricted probability model $\mathcal{M}(G_A)$ on the corpus f_A. Fortunately, this task can be easily solved. According to Theorem 1, the relative-frequency estimate on the corpus f_A is our unique solution. This yields for the rules $r \in G_A$

$$\hat{p}_A(r) = \frac{f_A(r)}{|f_A|} = \frac{f(r)}{\sum_{r \in G_A} f(r)}$$

Comparing these formulas to the formulas given in Definition 17, we conclude that for all non-terminal symbols A

$$\hat{p}_A(r) = p_{\mathcal{T}}(r) \quad \text{for all } r \in G_A$$

So, clearly, the treebank grammar $p_{\mathcal{T}}$ is a serious candidate for a maximum-likelihood estimate of the probability model $\mathcal{M}_G$ on $f_{\mathcal{T}}$. Now, as Chi and Geman (1998) verified, the treebank grammar $p_{\mathcal{T}}$ is indeed an instance of the probability model $\mathcal{M}_G$. So, combining all the results, it follows that

$$L(f_{\mathcal{T}};p_{\mathcal{T}}) = \max_{p \in \mathcal{M}_G} L(f_{\mathcal{T}};p)$$

Finally, since all $\hat{p}_A \in \mathcal{M}(G_A)$ are unique maximum-likelihood estimates, $p_{\mathcal{T}} \in \mathcal{M}_G$ is also a unique maximum-likelihood estimate **q.e.d.**

EM Training of PCFGs

Let us present first an overview of some theoretical work on using the EM algorithm for training of probabilistic context-free grammars.

- From 1966 to 1972, a group around Leonard E. Baum invents the forward-backward algorithm for probabilistic regular grammars (or hidden Markov models) and formally proves that this algorithm has some good convergence properties. See, for example, the overview presented in Baum (1972).
- Booth and Thompson (1973) discover a (still nameless) constraint for PCFGs. For PCFGs fulfilling the constraint, the probabilities of all full-parse trees sum to one.
- ◦ **Dempster et al. (1977) invent the EM algorithm.**

- Baker (1979) invents the inside-outside algorithm (as a generalization of the forward-backward algorithm). The training corpus of this algorithm, however, is not allowed to contain more than one single sentence.
- Lari and Young (1990) generalize Baker's inside-outside algorithm (or in other words, they invent the modern form of the inside-outside algorithm): The training corpus of their algorithm may contain arbitrary many sentences.
- Pereira and Schabes (1992) use the inside-outside algorithm for estimating a PCFG for English on a partially bracketed corpus.
- Sanchez and Benedi (1997) and Chi and Geman (1998) formally prove that for treebank grammars and grammars estimated by the EM algorithm, the probabilities of all full-parse trees sum to one.
- Prescher (2001) formally proves that the inside-outside algorithm is a dynamic programming instance of the EM algorithm for PCFGs. As a consequence, the inside-outside algorithm inherits the convergence properties of the EM algorithm (no formal proofs of these properties have been given by Baker and Lari and Young).
- Nederhof and Satta (2003) discover that the PCFG standard-constraints ("the probabilities of the rules with the same left-hand side are summing to one") are dispensable[12].

The overview presents two interesting streams. First, it suggests that the forward-backward algorithm is a special instance of the inside-outside algorithm, which in turn is a special instance of the EM algorithm. Second, it appears to be worthy to reflect on our notion of a standard probability model for context-free grammars (see the results of Booth and Thompson (1973) and Nederhof and Satta (2003)). Surely, both topics are very interesting — Here, however, we would like to limit our selfs and refer the interested reader to the papers mentioned above. The rest of this paper is dedicated to the pure non-dynamic EM algorithm for PCFGs. We would like to present its procedure and its properties, thereby motivating that the EM algorithm can be successfully used to train a manually written grammar on a plain text corpus.

As a first step, the following theorem shows that the EM algorithm is not only related to the inside-outside algorithm, but that the EM algorithm is also strongly connected with the treebank-training method on which we we have focused so far.

Theorem 11 *Let $< G, p_0 >$ be a probabilistic context-free grammar, where p_0 is an arbitrary starting instance of the probability model $\mathcal{M}_G$. Let $f : \mathcal{Y} \to \mathcal{R}$ be a non-empty and finite corpus of* ***sentences of*** *G (terminal strings that have at least one full-parse tree $x \in \mathcal{X}$). Then, applied to the PCFG $< G, p_0 >$ and the sentence corpus f, the EM Algorithm performs*

[12] In our terms, their result can be expressed as follows. Let G be a context-free grammar, and let $\mathcal{M}_G^*$ be the probability model that disregards the PCFG standard-constraints

$$\mathcal{M}_G^* = \left\{ p \in \mathcal{M}(\mathcal{X}) \;\middle|\; p(x) = \prod_{r \in G} p(r)^{f_r(x)} \right\}$$

Obviously, it follows then that $\mathcal{M}_G \subseteq \mathcal{M}_G^*$. Exploiting their Corollary 8, however, it follows somewhat surprisingly that both models are even equal: $\mathcal{M}_G^* = \mathcal{M}_G$. As a trivial consequence, a maximum-likelihood estimate of the standard probability model $\mathcal{M}_G$ (on a corpus of trees or on a corpus of sentences) is also a maximum-likelihood estimate of the probability model $\mathcal{M}_G^*$ — as well as the other way round.

the following procedure

(1) *for each* $i = 1,\ 2,\ 3,\ \ldots$ *do*
(2) $q := p_{i-1}$
(3) ***E-step (PCFGs):*** *generate the* ***treebank*** $f_{\mathcal{T}_q} : \mathcal{X} \to \mathcal{R}$ *defined by*

$$f_{\mathcal{T}_q}(x) \ := f(y) \cdot q(x|y) \quad \textit{where } y = \textit{yield}(x)$$

(4) ***M-step (PCFGs):*** *read off the* ***treebank grammar*** $< G, p_{\mathcal{T}_q} >$
(5) $p_i := p_{\mathcal{T}_q}$
(6) *end // for each* i
(7) *print* $p_0, p_1, p_2, p_3, \ldots$

Moreover, these EM re-estimates allocate the corpus f *a sequence of corpus probabilities that is monotonic increasing*

$$L(f; p_0) \leq L(f; p_1) \leq L(f; p_2) \leq L(f; p_3) \leq \ldots$$

Proof See Definition 13, and theorems 5 and 10.

Here is a toy example that exemplifies **how** the EM algorithm is applied to PCFGs. Remind first that we showed in Example 6 that a treebank grammar is (in principle) able to disambiguate correctly different forms of PP attachment. Remind also that we had to introduce some simple PP mark-up to achieve this result. Although we are choosing here a simpler example (so that the number of EM calculations is kept small), the example shall provide us with an intuition about **why** the EM algorithm is (in principle) able to learn "good" PCFGs. Again, the toy example presents a CFG having a PP-attachment ambiguity. This time, however, the training corpus is not made up of full-parse trees but of sentences of the grammar. Two types shall occur in the given corpus: Whereas the sentences of the first type are ambiguous, the sentences of the second type are not. Our simple goal is to demonstrate that the EM algorithm is able to learn from the unambiguous sentences how to disambiguate the ambiguous ones. It is exactly this feature that enables the EM algorithm to learn highly ambiguous grammars from real-world corpora: Although it is almost guaranteed in practice that sentences have on average thousands of analyzes, no sentence will hardly be completely unambiguous. Almost all sentences in a given corpus contain partly unambiguous information — represented for example in small sub-trees that the analyzes of the sentence share. By weighting the unambiguous information "high", and at the same time, by weighting the ambiguous information "low" (indifferently, uniformly or randomly), the EM algorithm might output a PCFG that learned something, namely, a PCFG being able to select for almost all sentences the single analysis that fits best the information which is hidden but nevertheless encoded in the corpus.

Example 8 *We shall consider an experiment in which a manually written CFG is estimated by the EM algorithm. We shall assume that the following corpus of 15 sentences is given to us*

$f(y)$	y
5	y_1
10	$y2$

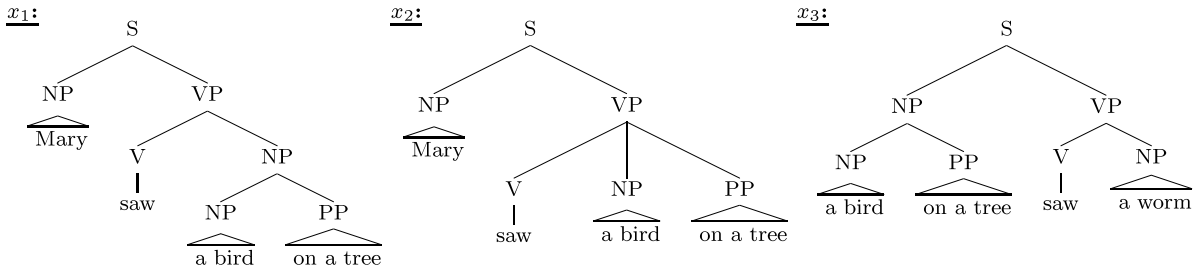
y_1 = *"Mary saw a bird on a tree"*
y_2 = *"a bird on a tree saw a worm"*

Using this text corpus, we shall compute the EM re-estimates of the following PCFG (that is able to parse all the corpus sentences)

$$
\begin{array}{lll}
S \longrightarrow NP\ VP & (1.00) \\
VP \longrightarrow V\ NP & (0.50) \\
VP \longrightarrow V\ NP\ PP & (0.50) \\
NP \longrightarrow NP\ PP & (0.25) \\
NP \longrightarrow Mary & (0.25) \\
NP \longrightarrow a\ bird & (0.25) \\
NP \longrightarrow a\ worm & (0.25) \\
PP \longrightarrow on\ a\ tree & (1.00) \\
V \longrightarrow saw & (1.00)
\end{array}
$$

Note: *As being displayed, the uniform distributions on the grammar fragments (G_S, G_{VP}, G_{NP} ...) serve in this example as a starting instance $p_0 \in \mathcal{M}_G$. On the one hand, this is not bad, since this is (or was) the common practice for EM training of probabilistic grammars. One the other hand, the EM algorithm gives us the freedom to experiment with a huge range of starting instances. Now, why not using the freedom that the EM algorithm donates? Indeed, the convergence proposition of Theorem 11 permits that some starting instances of the EM algorithm may lead to better re-estimates than other starting instances. So, clearly, if we are experimenting with more than one single starting instance, then we can seriously hope that our efforts are re-compensated by getting a better probabilistic grammar!*

First of all, note that the first sentence is ambiguous, whereas the second is not. The following figure displays the full-parse trees of both.



First EM iteration. In the **E-step** of the EM algorithm for PCFGs, a treebank $f_{\mathcal{T}_q}$ has to be generated on the basis of the starting instance $q := p_0$. The generation process is very easy. First, we have to calculate the probabilities of the full-parse trees x_1, x_2 and x_3, which in turn provide us with the probabilities of the sentences y_1 and y_2. Here are the results

$p_0(x)$	x
0.0078125	x_1
0.0312500	x_2
0.0078125	x_3

$p_0(y)$	y
0.0390625	y_1
0.0078125	y_2

For example, $p_0(x_1)$ and $p_0(y_1)$ are calculated as follows (using definitions 16 and 12.*):

$$p_0(x_1) \quad = \quad \prod_{r \in G} p(r)^{f_r(x_1)}$$

$$
\begin{aligned}
&= p(\text{S} \longrightarrow \text{NP VP}) \cdot p\left(\overset{\text{NP}}{\underset{\text{Mary}}{\triangle}}\right) \cdot p(\text{VP} \longrightarrow \text{V NP}) \cdot p(\text{V} \longrightarrow \text{saw}) \cdot p\left(\overset{\text{NP}}{\underset{\text{a bird}}{\triangle}}\right) \cdot p\left(\overset{\text{PP}}{\underset{\text{on a tree}}{\triangle}}\right) \\
&= 1.00 \cdot 0.25 \cdot 0.50 \cdot 1.00 \cdot 0.25 \cdot 0.25 \cdot 1.00 \\
&= 0.0078125
\end{aligned}
$$

$$
\begin{aligned}
p_0(y_1) &= \sum_{yield(x)=y_1} p(x) \\
&= p(x_1) + p(x_2) \\
&= 0.0078125 + 0.0312500 \\
&= 0.0390625
\end{aligned}
$$

Now, using the probability distribution $q := p_0$, we have to generate the treebank $f_{\mathcal{T}_q}$ by distributing the frequencies of the sentences to the full-parse trees. The result is

$f_{\mathcal{T}_q}(x)$	x
1	x_1
4	x_2
10	x_3

For example, $f_{\mathcal{T}_q}(x_1)$ is calculated as follows (see line (3) of the EM procedure in Theorem 11):

$$
\begin{aligned}
f_{\mathcal{T}_q}(x_1) &= f(\text{yield}(x_1)) \cdot q(x_1|\text{yield}(x_1)) \\
&= f(y_1) \cdot q(x_1|y_1) \\
&= f(y_1) \cdot \frac{q(x_1)}{q(y_1)} \\
&= 5 \cdot \frac{0.0078125}{0.0390625} \\
&= 1
\end{aligned}
$$

In the **E-step** of the EM algorithm for PCFGs, we have to read off the treebank grammar $< G, p_{\mathcal{T}_q} >$ from the treebank $f_{\mathcal{T}_q}$. Here is the result

S $\longrightarrow$ NP VP	(1.000)
VP $\longrightarrow$ V NP	($0.733 \approx \frac{1+10}{15}$)
VP $\longrightarrow$ V NP PP	($0.267 \approx \frac{4}{15}$)
NP $\longrightarrow$ NP PP	($0.268 \approx \frac{1+10}{41}$)
NP $\longrightarrow$ Mary	($0.122 \approx \frac{1+4}{41}$)
NP $\longrightarrow$ a bird	($0.366 \approx \frac{1+4+10}{41}$)
NP $\longrightarrow$ a worm	($0.244 \approx \frac{10}{41}$)
PP $\longrightarrow$ on a tree	(1.000)
V $\longrightarrow$ saw	(1.000)

The probabilities of the rules of this grammar form our first EM re-estimate p_1. So, we are ready for the second EM iteration. The following table displays the rules of our manually written CFG, as well as their probabilities allocated by the different EM re-estimates

CFG rule	p_0	p_1	p_2	p_3	...	p_{18}
S $\longrightarrow$ NP VP	1.000	1.000	1.000	1.000		1.000
VP $\longrightarrow$ V NP	0.500	0.733	0.807	0.850		0.967
VP $\longrightarrow$ V NP PP	0.500	0.267	0.193	0.150		0.033
NP $\longrightarrow$ NP PP	0.250	0.268	0.287	0.298		0.326
NP $\longrightarrow$ Mary	0.250	0.122	0.118	0.117		0.112
NP $\longrightarrow$ a bird	0.250	0.366	0.357	0.351		0.337
NP $\longrightarrow$ a worm	0.250	0.244	0.238	0.234		0.225
PP $\longrightarrow$ on a tree	1.000	1.000	1.000	1.000		1.000
V $\longrightarrow$ saw	1.000	1.000	1.000	1.000		1.000

In the 19^{th} iteration, nothing new happens. So, we can quit the algorithm and discuss the results. After all, of course, the most interesting thing for us is how these different PCFGs perform, if they are applied to ambiguity resolution. So, we will have a look at the probabilities these PCFGs allocate to the two analyzes of the first sentence. We have already noted several times that p prefers x_1 to x_2, if

$$p(\text{VP} \longrightarrow \text{V NP}) \cdot p(\text{NP} \longrightarrow \text{NP PP}) > p(\text{VP} \longrightarrow \text{V NP PP})$$

The following table displays the values of these terms for our re-estimated PCFGs

p	$p(\text{VP} \longrightarrow \text{V NP}) \cdot p(\text{NP} \longrightarrow \text{NP PP})$	$p(\text{VP} \longrightarrow \text{V NP PP})$
p_0	$0.500 \cdot 0.250 = 0.125$	0.500
p_1	$0.733 \cdot 0.268 = 0.196$	0.267
p_2	$0.807 \cdot 0.287 = \mathbf{0.232}$	0.193
p_3	$0.850 \cdot 0.298 = \mathbf{0.253}$	0.150
$\vdots$		
p_{18}	$0.967 \cdot 0.326 = \mathbf{0.315}$	0.033

So, the EM re-estimates prefer x_1 to x_2 starting with the second EM iteration, due to the fact that the term $p(\text{VP} \longrightarrow \text{V NP}) \cdot p(\text{NP} \longrightarrow \text{NP PP})$ is monotonic increasing (within a range from 0.125 to 0.315), while at the same time the term $p(\text{VP} \longrightarrow \text{V NP PP})$ is drastically monotonic decreasing (from 0.500 to 0.033).

Acknowledgments

Parts of the paper cover parts of the teaching material of two courses at ESSLLI 2003 in Vienna. One of them has been sponsored by the *European Chapter of the Association for Computational Linguistics (EACL)*, and both have been co-lectured by Khalil Sima'an and me. Various improvements of the paper have been suggested by Wietse Balkema, Gabriel Infante-Lopez, Karin Müller, Mark-Jan Nederhof, Breanndán Ó Nualláin, Khalil Sima'an, and Andreas Zollmann.

References

Backus, J. W. (1959). The syntax and semantics of the proposed international algebraic language of the Zürich ACM-GAMM Conference. In *Proceedings of the International Conference on Information Processing*, Paris.

Baker, J. K. (1979). Trainable grammars for speech recognition. In D. Klatt and J. Wolf (Eds.), *Speech Communication Papers for ASA'97*, pp. 547–550.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities III*, 1–8.

Booth, T. L. and R. A. Thompson (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers C-22*(5), 442–450.

Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02, Brown University.

Chi, Z. (1999). Statistical properties of probabilistic context-free grammars. *Computational Linguistics 25*(1).

Chi, Z. and S. Geman (1998). Squibs and discussions: Estimation of probabilistic context-free grammars. *Computational Linguistics 24*(2).

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*.

Cover, T. M. and J. A. Thomas (1991). *Elements of Information Theory*. New York: Wiley.

DeGroot, M. H. (1989). *Probability and statistics* (2 ed.). Addison-Wesley.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the *EM* algorithm. *J. Royal Statist. Soc. 39*(B), 1–38.

Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification — 2nd ed.* New York: Wiley.

Hopcroft, J. E. and J. D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review 106*, 620–630.

Johnson, M. (1998). PCFG models of linguistic tree representations. *Computational Linguistics 24*(4).

Klein, D. and C. D. Manning (2003). Accurate unlexicalized parsing. In *Proceedings of ACL-03*, Sapporo, Japan.

Lari, K. and S. J. Young (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language 4*, 35–56.

McLachlan, G. J. and T. Krishnan (1997). *The EM Algorithm and Extensions*. New York: Wiley.

Nederhof, M.-J. and G. Satta (2003). Probabilistic parsing as intersection. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT-03)*, Nancy, France.

Pereira, F. and Y. Schabes (1992). Inside-outside reestimation from partially bracketed corpora. In *Proceedings of ACL'92*, Newark, Delaware.

Prescher, D. (2001). Inside-outside estimation meets dynamic EM. In *Proceedings of IWPT-2001*, Beijing.

Prescher, D. (2002). *EM-basierte maschinelle Lernverfahren für natürliche Sprachen*. Ph. D. thesis, IMS, University of Stuttgart.

Ratnaparkhi, A. (1997). A simple introduction to maximum-entropy models for natural language processing. Technical report, University of Pennsylvania.

Sanchez, J. A. and J. M. Benedi (1997). Consistency of stochastic context-free grammars from probabilistic estimation based on growth transformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics 11*(1), 95–103.