Notes on Probabilistic Context-Free Grammars

11-761: Language and Statistics

March 18, 1999

1. Introduction

These notes describe the generalization of the forward-backward algorithm for hidden Markov models to a training algorithm for context-free grammars. Each of these training algorithms is special case of the EM algorithm, but the E-steps for them are quite different. In essence, the E-step for the forward-backward algorithm carries out dynamic programming on a trellis, and the E-step for the inside-outside algorithm carries out dynamic programming in a parse chart, which can be thought of as a kind of "3-dimensional trellis," or "tressel." The probabilities analogous to the forward and backward probabilities for parsing are called *inside* and *outside* probabilities, and so the generalized algorithm is called the *inside-outside algorithm*. This algorithm was first derived by Jim Baker, who did some of the important early work on the HMM approach to speech recognition at both CMU and IBM, and who is now head of *Dragon Systems*, one of the leading developers and marketers of speech technology.

2. The Basic Technique

To make a grammar probabilistic, we need to assign a probability to each context-free rewrite rule. But how should these probabilities be chosen? It is natural to expect that these probabilities should depend in some way on the domain that is being parsed. Just as a simple approach such as the trigram model used in speech recognition is "tuned" to a training corpus, we would like to "tune" our rule probabilities to our data. The statistical principle of maximum likelihood guides us in deciding how to proceed. According to this principle, we should choose our rule probabilities so as to maximize the likelihood of our data, whenever this is possible. In this section, we describe how this principle is realized in the Inside-Outside algorithm for parameter estimation of context-free grammars.

Let's begin by being concrete, and considering an example of a simple sentence and syntactic parse.

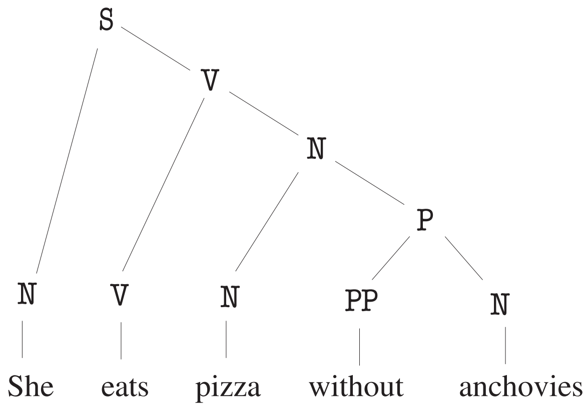

Figure 1

Implicit in this parse are four context-free rules of the form $\texttt{A} \to \texttt{B C}$. These are the rules

$$\texttt{S} \to \texttt{N V}$$
$$\texttt{V} \to \texttt{V N}$$
$$\texttt{N} \to \texttt{N P}$$
$$\texttt{P} \to \texttt{PP N}\,.$$

In addition, there are five "lexical productions" of the form $\texttt{A} \to \texttt{w}$:

$$\texttt{N} \to \text{She}$$
$$\texttt{V} \to \text{eats}$$
$$\texttt{N} \to \text{pizza}$$
$$\texttt{PP} \to \text{without}$$
$$\texttt{N} \to \text{anchovies.}$$

Of course, from a purely syntactic point-of-view, this sentence should also have a very different parse. To see this, just change the word "anchovies" to "hesitation":

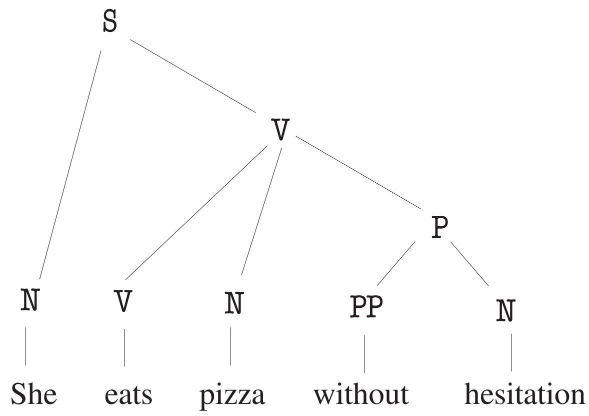

Figure 2

Here two new context-free rules have been used:

$$\texttt{V} \rightarrow \texttt{V N P}$$

$$\texttt{N} \rightarrow \text{hesitation}\,.$$

In the absence of other constraints, *both* parses are valid for each sentence. That is, one could, at least syntactically, speak of a type of "pizza without hesitation," though this would certainly be semantic gibberish, if not completely without taste. One of the goals of probabilistic training and parsing is to enable the statistics to correctly distinguish between such structures for a given sentence. This is indeed a formidable problem. Throughout this chapter we will refer to the above sentences and grammar to demonstrate how the training algorithm is actually carried out. While certainly a "toy" example, it will serve to illustrate the general algorithm, as well as to demonstrate the strengths and weaknesses of the approach.

Let's now assume that we have probabilities assigned to each of the above context-free rewrite rules. These probabilities will be written as, for example,

$$\phi(\texttt{S} \rightarrow \texttt{N V})$$

or as

$$\phi(\texttt{N} \rightarrow \text{pizza})$$

and we call these numbers the *parameters* of our probabilistic model. The only requirements of these numbers are that they be non-negative, and that for each nonterminal $\texttt{A}$ they sum to one; that is,

$$\sum_{\alpha} \phi(\texttt{A} \rightarrow \alpha) = 1$$

for each A, where the sum is over all strings of words and nonterminals that the grammar allows the symbol A to rewrite as. For our example grammar, this requirement is spelled out as follows:

$$\begin{aligned}
\phi(\mathtt{N} \to \mathtt{N\ P}) + \phi(\mathtt{N} \to \text{pizza}) + \phi(\mathtt{N} \to \text{anchovies}) &\quad + \\
+\, \phi(\mathtt{N} \to \text{hesitation}) + \phi(\mathtt{N} \to \text{She}) &= 1 \\
\phi(\mathtt{V} \to \mathtt{V\ N}) + \phi(\mathtt{V} \to \mathtt{V\ N\ P}) + \phi(\mathtt{V} \to \text{eats}) &= 1 \\
\phi(\mathtt{S} \to \mathtt{N\ V}) &= 1 \\
\phi(\mathtt{P} \to \mathtt{PP\ N}) &= 1 \\
\phi(\mathtt{PP} \to \text{without}) &= 1
\end{aligned}$$

So, the only parameters that are going to be trained, are those associated with rewriting a noun N or a verb V; all others are constrained to be equal to one.

The Inside-Outside algorithm starts from some initial setting of the parameters, and iteratively adjusts them so that the likelihood of the training corpus (in this case the two sentences “She eats pizza without anchovies” and “She eats pizza without hesitation”) increases. To write down the computation of this likelihood for our example, we'll have to introduce a bit of notation. First, we'll write

$$W_1 = \text{“She eats pizza without anchovies”}$$

and

$$W_2 = \text{“She eats pizza without hesitation”}.$$

Also, T_1 will refer to the parse in Figure 1, and T_2 will refer to the parse in Figure 2. Then the statistical model assigns probabilities to these parses as

$$\begin{aligned}
P_\phi(W_1, T_1) &= \phi(\mathtt{S} \to \mathtt{N\ V})\, \phi(\mathtt{V} \to \mathtt{V\ N})\, \phi(\mathtt{N} \to \mathtt{N\ P}) \;\times \\
&\times\;\; \phi(\mathtt{P} \to \mathtt{PP\ N})\, \phi(\mathtt{N} \to \text{She})\, \phi(\mathtt{V} \to \text{eats}) \;\times \\
&\times\;\; \phi(\mathtt{N} \to \text{pizza})\, \phi(\mathtt{PP} \to \text{without})\, \phi(\mathtt{N} \to \text{anchovies})
\end{aligned}$$

and

$$\begin{aligned}
P_\phi(W_2, T_1) &= \phi(\mathtt{S} \to \mathtt{N\ V})\, \phi(\mathtt{V} \to \mathtt{V\ N\ P})\, \phi(\mathtt{P} \to \mathtt{P\ PP}) \;\times \\
&\times\;\; \phi(\mathtt{N} \to \text{She})\, \phi(\mathtt{V} \to \text{eats})\, \phi(\mathtt{N} \to \text{pizza}) \times \\
&\times\;\; \phi(\mathtt{PP} \to \text{without})\, \phi(\mathtt{N} \to \text{hesitation})
\end{aligned}$$

In the absence of other restrictions, each sentence can have both parses. Thus we also have

$$\begin{aligned} P_\phi(W_1,T_2) &= \phi(\texttt{S} \to \texttt{N V})\,\phi(\texttt{V} \to \texttt{V N P})\,\phi(\texttt{P} \to \texttt{P PP}) \times \\ &\times \phi(\texttt{N} \to \text{She})\,\phi(\texttt{V} \to \text{eats})\,\phi(\texttt{N} \to \text{pizza}) \times \\ &\times \phi(\texttt{PP} \to \text{without})\,\phi(\texttt{N} \to \text{anchovies}) \end{aligned}$$

and

$$\begin{aligned} P_\phi(W_2,T_1) &= \phi(\texttt{S} \to \texttt{N V})\,\phi(\texttt{V} \to \texttt{V N})\,\phi(\texttt{N} \to \texttt{N P}) \times \\ &\times \phi(\texttt{P} \to \texttt{PP N})\,\phi(\texttt{N} \to \text{She})\,\phi(\texttt{V} \to \text{eats}) \times \\ &\times \phi(\texttt{N} \to \text{pizza})\,\phi(\texttt{PP} \to \text{without})\,\phi(\texttt{N} \to \text{hesitation}) \end{aligned}$$

The likelihood of our corpus with respect to the parameters ϕ is thus

$$L(\phi) = (P_\phi(W_1,T_1) + P_\phi(W_1,T_2))(P_\phi(W_2,T_1) + P_\phi(W_2,T_2).$$

In general, the probability of a sentence W is

$$P_\phi(W) = \sum_T P_\phi(W,T)$$

where the sum is over all valid parse trees that the grammar assigns to W, and if our training corpus comprises sentences $W_1, W_2, \ldots, W_N$, then the likelihood $L(\phi)$ of the corpus is given by

$$L(\phi) = P_\phi(W_1)P_\phi(W_2)\cdots P_\phi(W_N).$$

Starting at some initial parameters ϕ, the inside-algorithm reestimates the parameters to obtain new parameters ϕ' for which $L(\phi') \geq L(\phi)$. This process is repeated until the likelihood has converged. Since it is quite possible that starting with a different initialization of the parameters could lead to a significantly larger or smaller likelihood in the limit, we say that the Inside-Outside algorithm "locally maximizes" the likelihood of the training data.

The Inside-Outside algorithm is a special case of the EM algorithm [3] for maximum likelihood estimation of "hidden" models. We will derive formulas for updating the parameters, as well as describe how the CYK algorithm is used to actually compute those updates. We also derive these from the general EM algorithm directly.

3. The Parameter Updates

In this section we'll define some more notation, and then write down the updates for the parameters. This will be rather general, and the formulas may seem complex, but in the next section we will give a detailed description of how these updates are actually computed using our sample grammar.

To write down how the Inside-Outside algorithm updates the parameters, it is most convenient to assume that the grammar is in Chomsky normal form. It should be emphasized, however, that this is only a convenience. In the following section we will work through the Inside-Outside algorithm for our toy grammar, which is not, in fact, in Chomsky normal form. But we'll assume here that we have a general context-free grammar G all of whose rules are either of the form $\mathtt{A} \to \mathtt{B}\ \mathtt{C}$ for nonterminals $\mathtt{A}, \mathtt{B}, \mathtt{C}$, or of the form $\phi(\mathtt{A} \to \mathrm{w})$ for some word w.

Suppose that we have chosen values ϕ for our rule probabilities. Given these probabilities and a training corpus of sentences $W_1, W_2, \ldots, W_N$, the parameters are reestimated to obtain new parameters ϕ' as follows:

$$\phi'(\mathtt{A} \to \mathtt{B}\ \mathtt{C}) = \frac{\mathit{count}(\mathtt{A} \to \mathtt{B}\ \mathtt{C})}{\sum_\alpha \mathit{count}(\mathtt{A} \to \alpha)}$$

and

$$\phi'(\mathtt{A} \to \mathrm{w}) = \frac{\mathit{count}(\mathtt{A} \to \mathrm{w})}{\sum_\alpha \mathit{count}(\mathtt{A} \to \alpha)}$$

where

$$\mathit{count}(\mathtt{A} \to \mathtt{B}\ \mathtt{C}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \to \mathtt{B}\ \mathtt{C}, W_i)$$

and

$$\mathit{count}(\mathtt{A} \to \mathrm{w}) = \sum_{i=1}^{N} c_\phi(\mathtt{A} \to \mathrm{w}, W_i)$$

The number $c_\phi(\mathtt{A} \to \alpha, W_i)$ is the expected number of times that the rewrite rule $\mathtt{A} \to \alpha$ is used in generating the sentence W_i when the rule probabilities are given by ϕ. To give a formula for these expected counts, we need two more pieces of notation. The first piece of notation is standard in the automata literature (see, for example, [4]). If beginning with a nonterminal $\mathtt{A}$ we can derive a string γ of words and nonterminals by applying a sequence of rewrite rules from our grammar, then we write

$$\mathtt{A} \stackrel{*}{\Rightarrow} \gamma\,,$$

and say that $\mathtt{A}$ *derives* γ. So, if a sentence $W = w_1\, w_2\, \cdots\, w_n$ can be parsed by the grammar we can write

$$\mathtt{S} \stackrel{*}{\Rightarrow} w_1\, w_2\, \cdots\, w_n.$$

In the notation used above, the probability of the sentence given our probabilistic grammar is then

$$P_\phi(W) = \sum_T P_\phi(W,T) = P_\phi(\mathtt{S} \stackrel{*}{\Rightarrow} w_1\, w_2 \cdots w_n)\,.$$

The other piece of notation is just a shorthand for certain probabilities. The probability that the nonterminal $\mathtt{A}$ derives the string of words $w_i \cdots w_j$ in the sentence $W = w_1 \cdots w_n$ is denoted by $\alpha_{ij}(\mathtt{A})$. That is,

$$\alpha_{ij}(\mathtt{A}) = P_\phi(\mathtt{A} \stackrel{*}{\Rightarrow} w_i \cdots w_j)\,.$$

Also, we set the probability that beginning with the start symbol $\mathtt{S}$ we can derive the string $w_1 \cdots w_{i-1}\, \mathtt{A}\, w_{j+1} \cdots w_n$ equal to $\beta_{ij}(\mathtt{A})$. That is,

$$\beta_{ij}(\mathtt{A}) = P_\phi(\mathtt{S} \stackrel{*}{\Rightarrow} w_1 \cdots w_{i-1}\, \mathtt{A}\, w_{j+1} \cdots w_n)\,.$$

The alphas and betas are referred to, respectively, as *inside* and *outside* probabilities.

We are finally ready to give the formula for computing the expected counts. For a rule $\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C}$, the expected number of times that the rule is used in deriving the sentence W is

$$c_\phi(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C},\, W) \;=\; \frac{\phi(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C})}{P_\phi(W)} \sum_{1 \le i \le j \le k \le n} \beta_{ik}(\mathtt{A})\, \alpha_{ij}(\mathtt{B})\, \alpha_{j+1,k}(\mathtt{C})\,.$$

Similarly, the expected number of times that a lexical rule $\mathtt{A} \rightarrow \mathrm{w}$ is used in deriving W is given by

$$c_\phi(\mathtt{A} \rightarrow \mathrm{w},\, W) \;=\; \frac{\phi(\mathtt{A} \rightarrow \mathrm{w})}{P_\phi(W)} \sum_{1 \le n} \beta_{ii}(\mathtt{A})\,.$$

To actually carry out the computation, we need an efficient method for computing the α's and β's. Fortunately, there is an efficient way of computing these, based upon the following recurrence relations. If we still assume that our grammar is in Chomsky normal form, then it is easy to see that the α's must satisfy

$$\alpha_{ij}(\mathtt{A}) = \sum_{B,C} \sum_{i \le k \le j} \phi(\mathtt{A} \rightarrow \mathtt{B}\ \mathtt{C})\, \alpha_{ik}(\mathtt{B})\, \alpha_{k+1,j}(\mathtt{C})$$

for $i < j$ if we take

$$\alpha_{ii}(\mathtt{A}) = \phi(\mathtt{A} \rightarrow \mathrm{w}_{\mathtt{i}})\,.$$

Intuitively, this formula says that the inside probability $\alpha_{ij}(\mathtt{A})$ is computed as a sum over all possible ways of drawing the following picture:

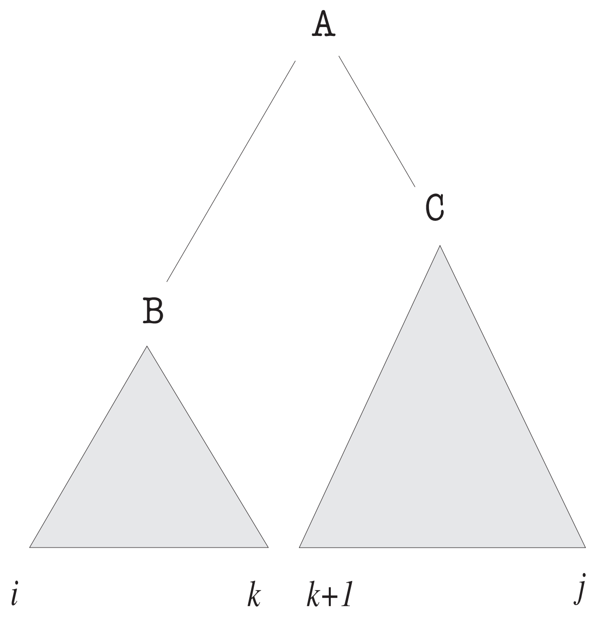

Figure 3

(Caveat Emptor: This formula and picture should be clear before reading on!)

In the same way, if the outside probabilities are initialized as $\beta_{1n}(\mathtt{S}) = 1$ and $\beta_{1n}(\mathtt{A}) = 0$ for $\mathtt{A} \neq \mathtt{S}$, then the β's are given by the following recursive expression:

$$\begin{aligned}\beta_{ij}(\mathtt{A}) &= \sum_{B,C} \sum_{1 \leq k < i} \phi(\mathtt{B} \rightarrow \mathtt{C}\ \mathtt{A})\ \alpha_{k,i-1}(\mathtt{C})\ \beta_{kj}(\mathtt{B}) + \\ &+ \sum_{B,C} \sum_{n \geq k > j} \phi(\mathtt{B} \rightarrow \mathtt{A}\ \mathtt{C})\ \alpha_{j+1,k}(\mathtt{C})\ \beta_{ik}(\mathtt{B}).\end{aligned}$$

Again, the first pair of sums can be viewed as considering all ways of drawing the following picture:

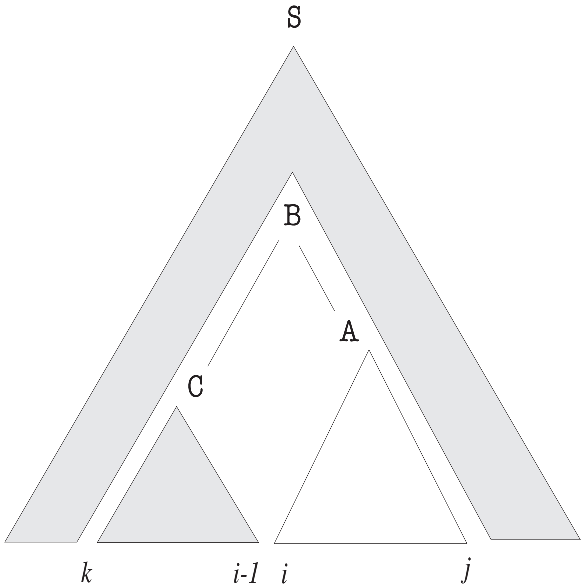

Figure 4

Together with the update formulas, the above recurrence formulas form the core of the Inside-Outside algorithm.

To summarize, the Inside-Outside algorithm consists of the following steps. First, chose some initial parameters ϕ and set all of the counts $count(\mathtt{A} \rightarrow \alpha)$ to zero. Then, for each sentence W_i in the training corpus, compute the inside probabilities α and the outside probabilities β. Then compute the expected number of times that each rule $\mathtt{A} \rightarrow \alpha$ is used in generating the sentence W_i. These are the numbers $c_\phi(\mathtt{A} \rightarrow \alpha, W_i)$. For each rule $\mathtt{A} \rightarrow \alpha$ add the number $c_\phi(\mathtt{A} \rightarrow \alpha, W_i)$ to the total count $count(\mathtt{A} \rightarrow \alpha)$ and proceed to the next sentence. After processing each sentence in this way, reestimate the parameters to obtain

$$\phi'(\mathtt{A} \rightarrow \alpha) = \frac{count(\mathtt{A} \rightarrow \alpha)}{\sum_\gamma count(\mathtt{A} \rightarrow \gamma)}.$$

Then, repeat the process all over again, setting $\phi = \phi'$, and computing the expected counts with respect to the new parameters.

How do we know when to stop? During each iteration, we compute the probability

$$P_\phi(W) = P_\phi(\mathtt{S} \stackrel{*}{\Rightarrow} w_1 \cdots w_n) = \alpha_{1n}(S)$$

of each sentence. This enables us to compute the likelihood,

$$L(\phi) = P_\phi(W_1)\, P_\phi(W_2) \cdots P_\phi(W_N),$$

or, better yet, the log likelihood

$$\mathrm{LL}(\phi) = \sum_{i=1}^{N} \log\, P_\phi(W_i)\,.$$

The Inside-Outside algorithm is guaranteed not to decrease the log likelihood; that is, $\mathrm{LL}(\phi') - \mathrm{LL}(\phi) \geq 0$. One may decide to stop whenever the change in log likelihood is sufficiently small. In our experience, this is typically after only a few iterations for a large natural language grammar.

In the next section, we will return to our toy example, and detail how these calculations are actually carried out. Whether the grammar is small or large, feature-based or in standard context-free form, the basic calculations are the same, and an understanding of them for the following example will enable you to implement the algorithm for your own grammar.

4. Calculation of the Inside and Outside Probabilities

The actual implementation of these computations is usually carried out with the help of the CYK algorithm [4]. This is a cubic recognition algorithm, and it proceeds as follows for our example probabilistic grammar. First, we need to put the grammar into Chomsky normal form. In fact, there is only one flagrant rule, $\mathtt{V} \to \mathtt{V}\ \mathtt{N}\ \mathtt{P}$, which we break up into two rules $\mathtt{N\text{-}P} \to \mathtt{N}\ \mathtt{P}$ and $\mathtt{V} \to \mathtt{V}\ \mathtt{N\text{-}P}$, introducing a new nonterminal $\mathtt{N\text{-}P}$. Notice that since there is only one $\mathtt{N\text{-}P}$ rule, the parameter $\phi(\mathtt{N\text{-}P} \to \mathtt{N}\ \mathtt{P})$ will be constrained to be one, so that

$$\phi(\mathtt{V} \to \mathtt{V}\ \mathtt{N\text{-}P})\ \phi(\mathtt{N\text{-}P} \to \mathtt{N}\ \mathtt{P}) = \phi(\mathtt{V} \to \mathtt{V}\ \mathtt{N\text{-}P})$$

is our estimate for $\phi(\mathtt{V} \to \mathtt{V}\ \mathtt{N}\ \mathtt{P})$. We now want to fill up the CYK chart for our first sentence, which is shown below.

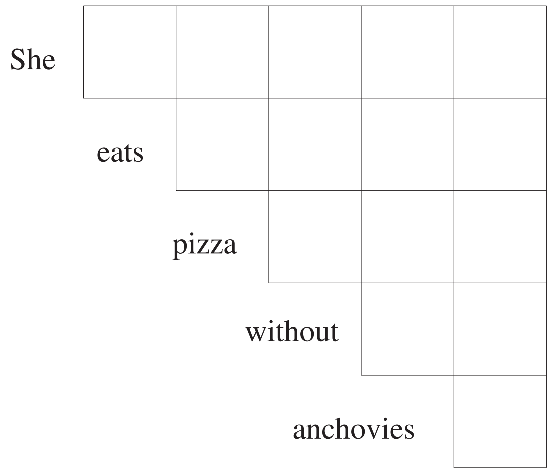

Figure 5

The algorithm proceeds by filling up the boxes in the order shown in the following picture.

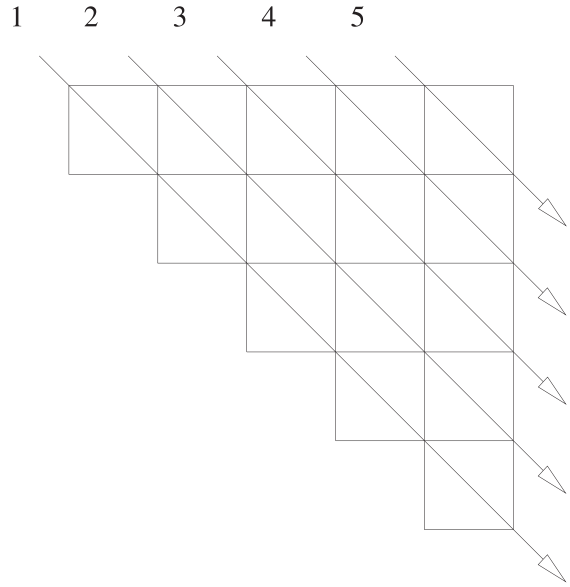

Figure 6

The first step is to fill the outermost diagonal. Into each box is entered the nonterminals which can generate the word associated with that box. Then the α's for the nonterminals which were entered are initialized. Thus, we obtain the chart

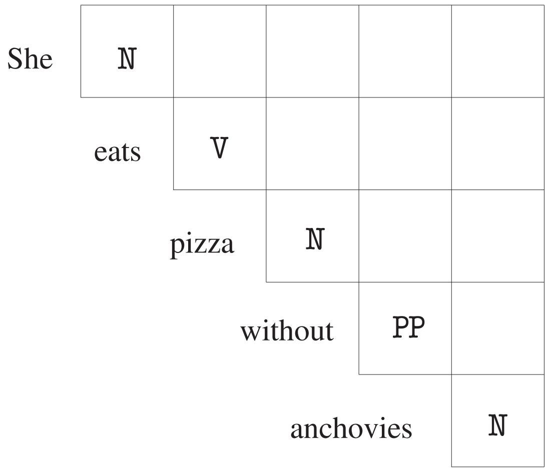

Figure 7

and we will have computed the inside probabilities

$$\alpha_{11}(\mathtt{N}) = \phi(\mathtt{N} \to \text{She}) \qquad \alpha_{22}(\mathtt{V}) = \phi(\mathtt{V} \to \text{eats})$$
$$\alpha_{33}(\mathtt{N}) = \phi(\mathtt{N} \to \text{pizza}) \qquad \alpha_{44}(\mathtt{PP}) = \phi(\mathtt{PP} \to \text{without})$$
$$\alpha_{55}(\mathtt{N}) = \phi(\mathtt{N} \to \text{anchovies})$$

All other α's are zero.

Now it happened in this case that each box contains only one nonterminal. If, however, a box contained two or more nonterminals, then the α for each nonterminal would be updated. Suppose, for example, that the word "anchovies" were replaced by "mushrooms." Then since "mushrooms" can be either a noun or a verb, the bottom part of the chart would appear as

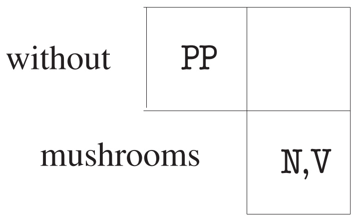

Figure 8

and we would have computed the inside probabilities

$$\alpha_{55}(\texttt{N}) = \phi(\texttt{N} \rightarrow \text{mushrooms})$$

and

$$\alpha_{55}(\texttt{V}) = \phi(\texttt{V} \rightarrow \text{mushrooms}).$$

In general, the rule for filling the boxes is that each nonterminal in box (i, j) must generate words $w_i \cdots w_j$, where the boxes are indexed as shown in the following picture.

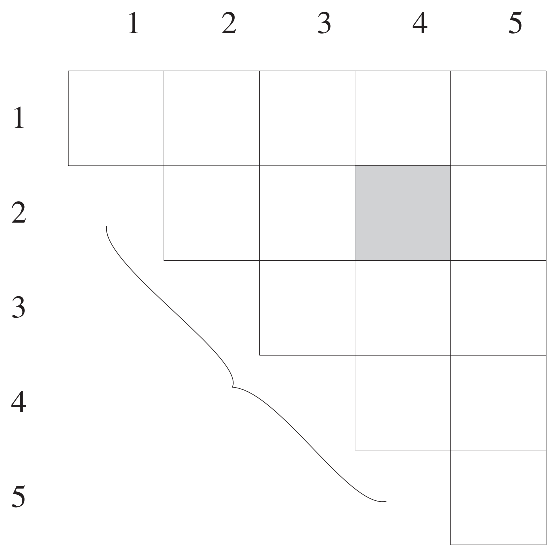

Figure 9

In this figure, the shaded box, which is box $(2, 4)$, spans words $w_2\, w_3\, w_4$.

To return now to our example, we proceed by filling the next diagonal and updating the α's, to get:

She	N	S			
eats		V	V		
pizza			N		
without				PP	P
anchovies					N

Figure 10

with

$$\begin{aligned}
\alpha_{12}(\mathtt{S}) &= \phi(\mathtt{S} \rightarrow \mathtt{N}\ \mathtt{V})\ \alpha_{11}(\mathtt{N})\ \alpha_{22}(\mathtt{V}) \\
\alpha_{23}(\mathtt{V}) &= \phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N})\ \alpha_{22}(\mathtt{V})\ \alpha_{33}(\mathtt{N}) \\
\alpha_{45}(\mathtt{P}) &= \phi(\mathtt{P} \rightarrow \mathtt{PP}\ \mathtt{N})\ \alpha_{44}(\mathtt{PP})\ \alpha_{55}(\mathtt{N})\,.
\end{aligned}$$

When we finish filling the chart, we will have

She	N	S	S		S
eats		V	V		V
pizza			N		N, N-P
without				PP	P
anchovies					N

Figure 11

with, for example, inside probabilities

$$\begin{aligned}\alpha_{25}(\mathtt{V}) &= \phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N})\ \alpha_{22}(\mathtt{V})\ \alpha_{35}(\mathtt{N}) + \\ &+ \phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N\text{-}P})\ \alpha_{22}(\mathtt{V})\ \alpha_{35}(\mathtt{N\text{-}P})\end{aligned}$$

and

$$\alpha_{15}(\mathtt{S}) = \phi(\mathtt{S} \rightarrow \mathtt{N}\ \mathtt{V})\ \alpha_{11}(\mathtt{N})\ \alpha_{25}(\mathtt{V})\,.$$

This last alpha is the total probability of the sentence:

$$\begin{aligned}\alpha_{15}(\mathtt{S}) &= P_\phi(\mathtt{S} \stackrel{*}{\Rightarrow} \text{She eats} \cdots \text{anchovies}) \\ &= \sum_T P_\phi(\text{She eats} \cdots \text{anchovies}, T).\end{aligned}$$

This completes the *inside pass* of the Inside-Outside algorithm. Now for the *outside pass*. The outside pass proceeds in the reverse order of Figure 6.

We initialize the β's by setting $\beta_{15}(\mathtt{S}) = 1$ and all other β's equal to zero. Then

$$\begin{aligned}\beta_{25}(\mathtt{V}) &= \phi(\mathtt{S} \rightarrow \mathtt{N}\ \mathtt{V})\ \alpha_{11}(\mathtt{N})\ \beta_{15}(\mathtt{S}) \\ \beta_{35}(\mathtt{N}) &= \phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N})\ \alpha_{22}(\mathtt{V})\ \beta_{25}(\mathtt{V}) \\ &\vdots \\ \beta_{55}(\mathtt{N}) &= \phi(\mathtt{P} \rightarrow \mathtt{PP}\ \mathtt{N})\ \alpha_{44}(\mathtt{PP})\ \beta_{45}(\mathtt{P})\,.\end{aligned}$$

The β's are computed in a top-down manner, retracing the steps that the inside pass took. For a given rule used in building the table, the outside probability for each child is updated using the outside probability of its parent, together with the inside probability of its sibling nonterminal.

Now we have all the necessary ingredients necessary to compute the counts. As an example of how these are computed, we have

$$\begin{aligned}c_\phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N}, W_1) &= \frac{\phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N})}{\alpha_{15}(\mathtt{S})}\,(\alpha_{22}(\mathtt{V})\ \alpha_{33}(\mathtt{N})\ \beta_{23}(\mathtt{V}) + \\ &+ \alpha_{22}(\mathtt{V})\ \alpha_{35}(\mathtt{N})\ \beta_{25}(\mathtt{V})) \\ &= \frac{\phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N})}{\alpha_{15}(\mathtt{S})}\,(\alpha_{22}(\mathtt{V})\ \alpha_{33}(\mathtt{N})\ \beta_{23}(\mathtt{V}))\end{aligned}$$

since $\beta_{23}(\mathtt{V}) = 0$. Also,

$$c_\phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N\text{-}P}, W_1) = \frac{\phi(\mathtt{V} \rightarrow \mathtt{V}\ \mathtt{N\text{-}P})}{\alpha_{15}(\mathtt{S})}\,(\alpha_{22}(\mathtt{V})\ \alpha_{35}(\mathtt{N\text{-}P})\ \beta_{25}(\mathtt{V}))\ .$$

We now proceed to the next sentence, "She eats pizza without hesitation." In this case, the computation proceeds exactly as before, except, of course, that all inside probabilities involving the probability $\phi(\texttt{N} \to \text{anchovies})$ are replaced by the probability $\phi(\texttt{N} \to \text{hesitation})$. When we are finished updating the counts for this sentence, the probabilities are recomputed as, for example,

$$\phi'(\texttt{V} \to \texttt{V N}) =$$
$$\frac{c_\phi(\texttt{V} \to \texttt{V N}, W_1) + c_\phi(\texttt{V} \to \texttt{V N}, W_2)}{\sum_{i=1,2} c_\phi(\texttt{V} \to \texttt{V N P}, W_i) + c_\phi(\texttt{V} \to \text{eats}, W_i) + c_\phi(\texttt{V} \to \texttt{V N}, W_i)}$$

$$\phi'(\texttt{V} \to \text{eats}) =$$
$$\frac{c_\phi(\texttt{V} \to \text{eats}, W_1) + c_\phi(\texttt{V} \to \text{eats}, W_2)}{\sum_{i=1,2} c_\phi(\texttt{V} \to \texttt{V N P}, W_i) + c_\phi(\texttt{V} \to \text{eats}, W_i) + c_\phi(\texttt{V} \to \texttt{V N}, W_i)}$$

Though we have described an algorithm for training the grammar probabilities, it is a simple matter to modify the algorithm to extract the most probable parse for a given sentence. For historical reasons, this is called the *Viterbi algorithm*, and the most probable parse is called the *Viterbi parse*. Briefly, the algorithm proceeds as follows. For each nonterminal $\texttt{A}$ added to a box (i, j), in the chart, we keep a record of which is the most probable way of rewriting $\texttt{A}$. That is, we determine which nonterminals $\texttt{B}, \texttt{C}$ and index k maximize the probability

$$\phi(\texttt{A} \to \texttt{B C})\ \alpha_{ik}(\texttt{B})\ \alpha_{k+1,j}(\texttt{C})\,.$$

When we reach the topmost nonterminal $\texttt{S}$, we can then "traceback" to construct the Viterbi parse. We shall leave the details of this algorithm as an exercise for the reader.

The computation outlined here has all of the essential features of the Inside-Outside algorithm applied to a "serious" natural language grammar. In the following section, we'll present some of the technical details.

5. Relation to the EM Algorithm

In this section we give a more technical demonstration of the convergence of the inside-outside algorithm, by formulating it directly as an EM algorithm. Our notation will follow the previous handout on the EM algorithm.

We'll now be a little more formal. Let G be a context-free grammar consisting of a collection of rules $\{A \to \alpha\}$, where each α is a string of terminals and nonterminals. For each string

$w \in \mathcal{L}(G)$, the language of G, there is a corresponding set of parse trees t, each of which has $w = w_1 \, w_2 \, \cdots \, w_N$ as leaves. If we observe only w, then for an ambiguous grammar, the actual tree used to derive w is hidden.

Suppose we have a joint distribution $P_\phi(w, t)$, giving the probability of deriving w using the tree t. Then the marginal distribution

$$P_\phi(w) = \sum_t P_\phi(w, t)$$

gives a language model. In the notation of Section 2, $P_\phi(w, t)$ is the *complete data density* $f_\phi(x)$ and $P_\phi(w)$ is the *incomplete data density* $g_\phi(y)$. The fiber $\mathcal{X}(w)$ over the sentence w is a finite collection of parse trees. The joint distribution takes the form

$$\begin{aligned} P_\phi(w, t) &= \prod_\omega \phi(\omega)^{c(\omega;\, t, w)} \\ &= \prod_{A \to \alpha} \phi(A \to \alpha)^{c(A \to \alpha;\, t, w)} \end{aligned}$$

where $c(A \to \alpha;\, t, w)$ is the number of times that the rule $A \to \alpha$ appears in the parse tree t for the sentence w. The parameters $\phi(A \to \alpha)$ are normalized so that

$$\sum_\alpha \phi(A \to \alpha) = 1\,.$$

Thus, there will be a Lagrange multiplier for each nonterminal A.

Such a model may be *deficient*, and not assign probability one to finite strings. A sufficient condition that this does not happen can be expressed by indexing the nonterminals as $A_1, \ldots, A_N$, and letting M be the $N \times N$ matrix given by

$$M_{ij} = \sum_\alpha \phi(A_i \to \alpha)\, n_j(\alpha)$$

where $n_j(\alpha)$ is the number of nonterminal symbols A_j appearing in α. If M has largest eigenvalue $\rho < 1$, then the language model $P_\phi(w)$ assigns probability one to finite sentences in the language of the grammar.

The model is parameterized by making the *Markov assumption* that the probability with which a nonterminal is rewritten as a string α depends only on the nonterminal, and not on any surrounding context. This assumption leads to an efficient training algorithm.

There are two distinct problems associated with this setup. The first, called the *language modeling problem*, is to find the set of parameters which maximize the probability

$\prod_{w \in \mathcal{C}} P_\phi(w)$ of some training corpus $\mathcal{C}$. The second, called the *parsing problem*, is to maximize the "correctness" of the most probable parse

$$\hat{t}(w) = \text{argmax}_t \, P_\phi(t \mid w)\,.$$

The EM algorithm is directly involved with only the language modeling problem. Experience has shown it to be difficult to couple the two problems.

To apply the EM algorithm, we consider the auxiliary function

$$\widehat{Q}(\phi' \mid \phi) = \sum_w c(w) \sum_t P_\phi(t \mid w) \log \frac{P_{\phi'}(t,w)}{P_\phi(t,w)}\,.$$

Taking the derivative $\partial/\partial\phi'(A \to \alpha)$ gives

$$\frac{\partial \widehat{Q}(\phi' \mid \phi)}{\partial \phi'(A \to \alpha)} = \sum_w c(w) \sum_t \frac{P_\phi(t \mid w) c(A \to \alpha;\, t, w)}{\phi'(A \to \alpha)}\,.$$

We thus need to compute the expected counts

$$\sum_t P_\phi(t \mid w) c(A \to \alpha;\, t, w)\,.$$

The sum $\sum_t$ is potentially exponential. But this is the same as evaluating

$$\frac{\phi(A \to \alpha)}{P_\phi(w)} \frac{\partial P_\phi(w)}{\partial \phi(A \to \alpha)} = \sum_t P_\phi(t \mid w) c(A \to \alpha;\, t, w)\,,$$

and it turns out that there is an efficient way of computing the partial derivative on the lefthand side.

We'll now assume, but only for convenience, that the grammar is in Chomsky normal form. Thus, each rule is either of the form $A \to BC$ or $A \to w$. The *position* of a rule $A \to BC$ within a tree t can be specified by a triple (i, j, k), $i \le j \le k$.

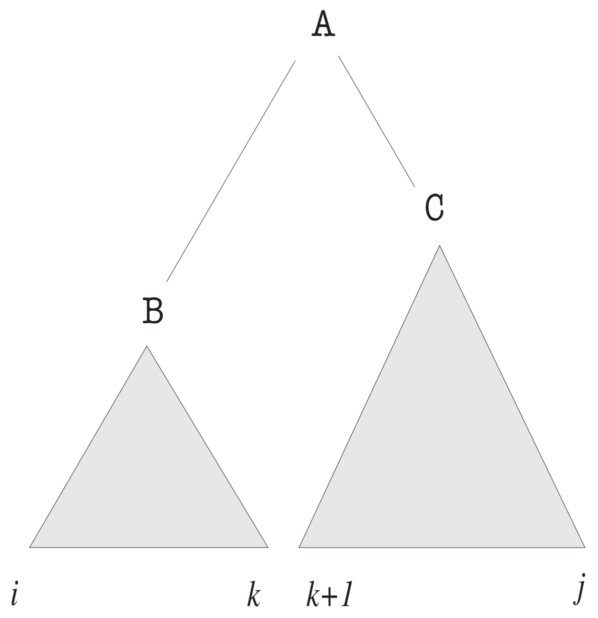

Figure 12

The partial derivative of the probability $P_\phi(S \Rightarrow w) = P_\phi(w)$ with respect to the parameter $\phi(A \to BC)$ only involves those parse trees which use the rule $A \to BC$. Consider the event "$S \Rightarrow w$ using $A \to BC$ in position (i,j,k)". Because of the Markov property, the probability of this event can be written as a product of four terms as follows:

$$\begin{aligned}&P_\phi(S \Rightarrow w;\ \text{using } A \to BC \text{ in position } (i,j,k)) = \\ &\quad P_\phi(S \Rightarrow w_1 \cdots w_{i-1}\, A\, w_{k+1} \cdots w_N) \times \\ &\quad\quad \times \phi(A \to BC)\, P(B \Rightarrow w_i \cdots w_j) P(c \Rightarrow w_{j+1} \cdots w_k)\,.\end{aligned}$$

From this it is not difficult to see that

$$\frac{\partial P_\phi(S \Rightarrow w)}{\partial \phi(A \to BC)} = \sum_{i,j,k} P_\phi(S \Rightarrow w_1 \cdots w_{i-1}\, A\, w_{k+1} \cdots w_N)\, P(B \Rightarrow w_i \cdots w_j)\, P(C \Rightarrow w_{j+1} \cdots w_k)\,.$$

Thus, the expected number of times that the rule $A \to BC$ is used in generating the sentence w using the model ϕ is given by

$$\begin{aligned} E_\phi[c(A \to BC; w)] &= \sum_t P_\phi(t \mid w) c(A \to BC;\, t, w) \\ &= \frac{\phi(A \to BC)}{P_\phi(w)} \sum_{i,j,k} \beta_{ik}(A) \alpha_{ij}(B) \alpha_{j+1k}(C) \end{aligned}$$

where

$$\alpha_{ij}(A) = P_\phi(A \Rightarrow w_i \cdots w_j)$$

and

$$\beta_{ij}(A) = P_\phi(S \Rightarrow w_1 \cdots w_{i-1}\, A\, w_{j+1} \cdots w_N)\,.$$

Similarly,

$$E_\phi[A \to a; w] = \frac{\phi(A \to a)}{P_\phi(w)} \sum_i \delta_a(w_i)\, \beta_{ii}(A)\,.$$

There is an efficient method for computing the α's and β's using the CKY chart-parsing algorithm. The method for doing this is implicit in the following formulas:

$$\begin{aligned}
\alpha_{ij}(A) &= \sum_{B,C} \sum_{i \le k \le j} \phi(A \to BC) \alpha_{ik}(B)\, \alpha_{k+1j}(C) \\
\alpha_{ii}(A) &= \phi(A \to w_i) \\
\beta_{ij}(A) &= \sum_{B,C} \sum_{k<i} \phi(B \to CA) \alpha_{ki-1}(C)\, \beta_{kj}(B) + \\
&+ \sum_{B,C} \sum_{k>j} \phi(B \to AC) \alpha_{j+1k}(C)\, \beta_{ik}(B) \\
\beta_{1N}(A) &= \delta_S(A)\,.
\end{aligned}$$

The reestimated parameters are then the normalized counts:

$$\phi'(A \to \alpha) = \lambda_A^{-1} \mathrm{count}(A \to \alpha)$$

where

$$\mathrm{count}(A \to \alpha) = \sum_{w \,\in\, \mathbf{e}} E_\phi[c(A \to \alpha); w]\,.$$

The sum is over all sentences in the training data $\mathbf{e}$. The Lagrange multiplier λ_A is given by

$$\lambda_A = \sum_\alpha \mathrm{count}(A \to \alpha)\,.$$

The results of Sections 2 and 3 prove that the resulting model does not assign a smaller likelihood to $\mathbf{e}$.

References

1. J.K. Baker, *Trainable grammars for speech recognition*, Proceedings of the Spring Conference of the Acoustical Society of America (Boston, MA), pp. 547–550, 1979.

2. L.E. Baum, *An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process*, Inequalities **3** (1972), 1–8.

3. A.P. Dempster, N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistical Society **39** (1977) B, 1–38.

4. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

5. F. Jelinek, J. Lafferty, and R. Mercer. *Basic methods of probabilistic context-free grammars.* IBM Research Report RC 16374, 1991.

6. F. Jelinek, J. Lafferty, and R. Mercer. *Basic methods of probabilistic context-free grammars.* In *Speech Recognition and Understanding: Recent Advances, Trends, and Applications,* P. Laface and R. De Mori, editors. Springer Verlag, Series F: Computer and Systems Sciences, Volume 75, 1992.

7. J. D. Lafferty, *Statistical training of grammars*, in *Statistically-Based Computer Analysis of English*, E. Black and G. Leech editors, Language and Computers Series, Radopi Publishers, Amsterdam, 1993.