Prefix Probabilities for Linear Indexed Grammars

Mark-Jan Nederhof
DFKI
Saarbrucken, Germany
nederhof@dfki.de

Anoop Sarkar
Dept. of Computer and Information Science
University of Pennsylvania
anoop@linc.cis.upenn.edu

Giorgio Satta
Dip. di Elettronica e Informatica
Università di Padova
satta@dei.unipd.it

Abstract

We show how prefix probabilities can be computed for stochastic linear indexed grammars (SLIGs). Our results apply as well to stochastic tree-adjoining grammars (STAGs), due to their equivalence to SLIGs.

1 Introduction

The problem of computing prefix probabilities for stochastic context-free languages is defined as follows. Given a word sequence $a_1 \cdots a_n$ over some alphabet Σ, which we call the input prefix, we must compute quantity $\sum_{w \in \Sigma^*} \Pr(a_1 \cdots a_n w)$. This problem has been discussed in [1, 4] with the main motivation of applications in speech recognition, where we are given some word sequence $a_1 \cdots a_{n-1}$, and must hypothesize the next word a_n.

The main idea leading to the solution of this problem is that all parts of context-free derivations that are potentially of unbounded size are captured into a set of equations that can be solved "off-line", i.e., before a specific prefix is considered. This is possible because the involved derivations do not depend on the given prefix. Once these equations have been solved, the results are stored. When computing the prefix probability for an actual input string, all possible derivations are then considered and a probability is computed, but for certain parts of these derivations the results that were computed off-line are used, in such a way that the computation is guaranteed to terminate.

Cases of derivations of potentially unbounded size might arise because of so called *unit rules*, i.e., rules of the form $A \to B$. Such rules potentially cause the grammar to be cyclic, which means that $A \to^* A$ might hold for some nonterminal A. This allows certain strings to have derivations of unbounded size. However, also a rule of e.g. the form $A \to Ba$ may effectively behave like a unit rule if a contributes to the unknown suffix following the actual input that is considered as prefix.

For stochastic tree-adjoining grammars (STAGs) similar problems arise. STAGs that are well-behaved and allow a bounded number of derivations for each complete sentence may require an unbounded number of derivations to be considered, once the input is regarded as a prefix followed by a suffix of unbounded length. The key idea to solving this problem is again to break up derivations into parts that are of potentially unbounded size and are

independent on actual input, and parts that are always of bounded length and do depend on input symbols. The probabilities of the former subderivations can be computed off-line, and the results are combined with subderivations of the latter kind during computation of the prefix probability for a given string.

The distinction between the two kinds of subderivations requires a certain notational system that is difficult to define for tree-adjoining grammars. We will therefore concentrate on stochastic linear indexed grammars instead, relying on their equivalence to STAGs [3]. The solution proposed in the present paper is an alternative to a different approach by the same authors in [2]. In that publication, a set of equations is transformed in order to distinguish off-line and on-line computations.

2 Computation of prefix probabilities

We refer the reader to [2] for the definition of LIG. In what follows, we use $\alpha, \beta, \ldots$ to denote strings of nonterminals associated with empty stacks of indices, $x, y, v, w, z, \ldots$ to denote strings of terminal symbols, and a to denote a terminal symbol. Without loss of generality we require that rules are of the form $A[\eta \circ\circ\,] \rightarrow \alpha\ B[\eta' \circ\circ\,]\ \beta$ with $|\eta\eta'| = 1$, or of the form $A[\,] \rightarrow z$, where $|z| \leq 1$.

As usual, $\rightarrow$ is extended to a binary relation between sentential forms, and its transitive and reflexive closure is denoted by $\rightarrow^*$. When we write $A[\sigma] \rightarrow^* \alpha\ B[\tau]\ \beta$, the indicated occurrence of $B[\tau]$ is the symbol that inherits the stack content of $A[\sigma]$ in the derivation, which we will call the *distinguished descendant* of $A[\sigma]$. We extend this notation to $A[\sigma] \rightarrow^* \alpha\ a\ \beta$, when a is generated in one step from the distinguished descendant of $A[\sigma]$ in a previous sentential form.

We first introduce a subrelation of $\rightarrow^*$ defined by $A[\sigma] \Rightarrow^* \varepsilon$ if $A[\sigma] \rightarrow^* \varepsilon$, and $A[\sigma] \Rightarrow^* B[\tau]$ if $A[\sigma] \rightarrow^* B[\tau]$ and this derivation does not end on a subderivation of the form $C[\tau] \rightarrow^+ B[\tau]$, for any C, where no elements that belong to τ are popped and pushed again. When we write $A[\sigma] \Rightarrow^* X$, then X is of the form $B[\,]$ or ε.

Based on this, two further subrelations of relation $\rightarrow^*$, written $\rightarrow^*_{ver}$ and $\rightarrow^*_{hor}$, are defined below by means of deduction steps. The distinction between $\rightarrow^*_{ver}$ and $\rightarrow^*_{hor}$ is made in order to record how derivations were built up from subderivations. In the case of $\rightarrow^*_{hor}$, the derivation was constructed from two subderivations $A[\,] \rightarrow^* v\ B[\,]\ w$ and $B[\,] \rightarrow^* x\ C[\,]\ y$. In all other cases, we use $\rightarrow^*_{ver}$. This distinction is needed to avoid spurious ambiguity in applications of the deduction steps: the result from combining $A[\,] \rightarrow^* v\ B[\,]\ w$ and $B[\,] \rightarrow^* x\ C[\,]\ y$, viz. $A[\,] \rightarrow^*_{hor} v\ x\ C[\,]\ y\ w$, is not allowed to combine with a third subderivation $C[\,] \rightarrow^* z\ D[\,]\ q$. Note that the desired derivation $A[\,] \rightarrow^*_{hor} v\ x\ z\ D[\,]\ q\ y\ w$ can be derived by combining $B[\,] \rightarrow^* x\ C[\,]\ y$ and $C[\,] \rightarrow^* z\ D[\,]\ q$, and then $A[\,] \rightarrow^* v\ B[\,]\ w$ with the result of this.

$$\frac{}{\varepsilon \rightarrow^*_{ver} \varepsilon} \qquad (1) \qquad \frac{A[\,] \rightarrow^*_{ver} v \quad \alpha \rightarrow^*_{ver} w \quad \alpha \neq \varepsilon}{A[\,\alpha] \rightarrow^*_{ver} v\ w} \qquad (2)$$

$$\frac{A[\,] \rightarrow^* a}{A[\,] \rightarrow^*_{ver} a} \qquad (3) \qquad \frac{\begin{array}{ll} A[\,] \rightarrow^* B[\sigma] & \alpha \rightarrow^*_{ver} v_\alpha \\ B[\circ\circ\,] \rightarrow \alpha\ C[p \circ\circ\,]\ \beta & \beta \rightarrow^*_{ver} v_\beta \\ C[\,] \rightarrow^* D[\,] & \gamma \rightarrow^*_{ver} v_\gamma \\ D[p \circ\circ\,] \rightarrow \gamma\ E[\circ\circ\,]\ \delta & \delta \rightarrow^*_{ver} v_\delta \\ E[\sigma] \Rightarrow^* X & v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v_\alpha\ v_\gamma\ X\ v_\delta\ v_\beta} \qquad (4)$$

$$\frac{\begin{array}{ll} A[\,] \rightarrow^* B[\sigma] & lab \in \{ver, hor\} \\ B[\circ\circ] \rightarrow \alpha\ C[p\circ\circ]\ \beta & \alpha \rightarrow^*_{ver} v_\alpha \\ C[\,] \rightarrow^*_{lab} v\ D[\,]\ w & \beta \rightarrow^*_{ver} v_\beta \\ D[\,] \rightarrow^* E[\,] & \gamma \rightarrow^*_{ver} v_\gamma \\ E[p\circ\circ] \rightarrow \gamma\ F[\circ\circ]\ \delta & \delta \rightarrow^*_{ver} v_\delta \\ F[\sigma] \Rightarrow^* X & v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v_\alpha\ v\ v_\gamma\ X\ v_\delta\ w\ \ v_\beta} \quad (5)$$

$$\frac{\begin{array}{l} A[\,] \rightarrow^*_{ver} v\ B[\,]\ w \\ B[\,] \rightarrow^*_{lab} x\ C[\,]\ y \quad lab \in \{ver, hor\} \end{array}}{A[\,] \rightarrow^*_{hor} v\ x\ C[\,]\ y\ w} \quad (6)$$

$$\frac{\begin{array}{l} A[\,] \rightarrow^*_{ver} v\ B[\,]\ w \\ B[\,] \rightarrow^*_{ver} x \end{array}}{A[\,] \rightarrow^*_{ver} v\ x\ w} \quad (7)$$

$$\frac{\begin{array}{l} A[\,] \Rightarrow^* B[\sigma] \\ B[\,] \rightarrow^*_{hor} v\ C[\,]\ w \\ C[\sigma] \Rightarrow^* X \qquad \sigma \neq \varepsilon \end{array}}{A[\,] \rightarrow^*_{ver} v\ X\ w} \quad (8)$$

We now discuss how LIG derivations are uniquely partitioned into subderivations by the above steps. We will explain later how the above steps can be used in the computation of prefix probabilities. We call *spine* any path in the parse tree that leads from a node that is not a distinguished child of its father (or that does not have a father, in the case of the root), down to a leaf following distinguished children. This means that for an instance of a rule $A[\eta\circ\circ] \rightarrow \alpha\ B[\eta'\circ\circ]\ \beta$ in the parse tree, the nodes corresponding to symbols in α and β are each the first node of a distinct spine. Also, the spine belonging to the node which corresponds to $A[\eta\circ\circ]$ leads down along the node corresponding to $B[\eta'\circ\circ]$. At both ends of a spine, the stack of indices associated with the nonterminals is empty. In between, the height of the stack may alternately grow and shrink. This is shown in Figure 1. The horizontal axis represents nodes along the spine, and the vertical axis represents the height of the stack.

At some instances of rules, non-empty input is found at some child of a node on the spine that does itself not belong to the spine. We always investigate such rules in pairs: if one rule pushes p on the stack, we locate the unique rule that pops that p; only one of the two rules needs to be associated with non-empty input. Three instances of such pairs of rules are indicated in the figure.

In Figure 1, the parts of the spine labelled by a and b are accounted for by step (4). From these two parts, the part labelled c is obtained through step (6). This step combines paths

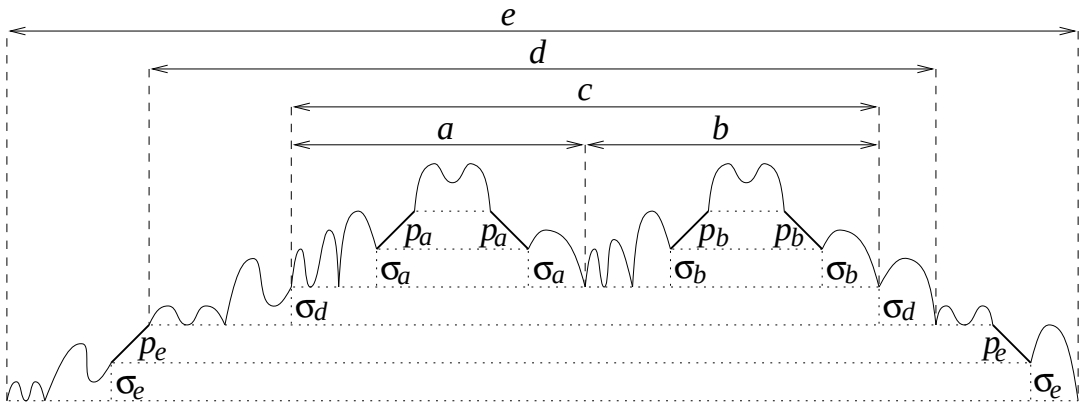

Figure 1: Development of the stack along a spine, and partitioning according to deduction steps.

in a "horizontal" way, hence the label *hor* in the consequent. The path is then extended to the path d in a vertical way by applying step (8). Again vertically, step (5) extends the path to path e by identifying one more pair of rules where non-empty input is found.

Each stack development along a spine, exemplified by Figure 1, can be partitioned in exactly one way according to the deduction steps. The proof of this fact is rather involved and is not reported in this long abstract.

We can now discuss how to compute prefix probabilities using steps (1) to (8). We can compute the inside probability of a given string w by applying the deduction steps in reverse for the relation $S[\,] \rightarrow^*_{ver} w$. This gives rise to a unique partitioning into subderivations for each possible derivation of w in the grammar. We multiply the probabilities attached to the rules that are used in the derivations, and we add probabilities where more than one derivation exists due to ambiguity.

We see that statements of the form $C[\,] \rightarrow^* D[\,]$ in e.g. step (4) and $A[\,] \rightarrow^* a$ in step (3) cannot themselves be derived by the deduction steps. It is assumed the probabilities of such derivations are computed off-line, which is possible since they do not depend on actual input. Also, the joint probability of the pair of derivations $A[\,] \rightarrow^* B[\sigma]$ and $E[\sigma] \Rightarrow^* X$ in step (4) can be precomputed for a given combination of A, B, E, and X, even though there may be an infinite number of stacks σ. These off-line computations can be carried out by solving systems of equations that express recursive relations among probabilities of derivations. Again, due to space limitations these systems will not be introduced in this long abstract.

It is easy to see that the backward application of the deduction steps must necessarily terminate. This is independent of whether a LIG allows infinite ambiguity.

If prefix probabilities are to be computed instead of inside probabilities, the deduction steps need to be slightly altered. For example, the condition $v_\alpha v_\beta v_\gamma v_\delta \neq \varepsilon$ in step (4) needs to be reformulated to the effect that at least one symbol from $v_\alpha v_\beta v_\gamma v_\delta$ should belong to the input, i.e. the prefix. Further, probabilities of derivations of the form $A[\,] \rightarrow^* B[\,]\, w$ should be computed off-line, where w belongs to the unknown suffix. (Cf. unit rules and rules of the form $A \rightarrow Ba$ in the case of context-free grammars.)

It is easy to see that the deduction steps are consistent, in the sense that $\alpha \rightarrow^*_{ver} \beta$ or $\alpha \rightarrow^*_{hor} \beta$ implies $\alpha \rightarrow^* \beta$. That the deduction steps are also complete, i.e., that $A[\,] \rightarrow^*_{ver} w$ can be derived if $A[\,] \rightarrow^* w$, is more difficult to show and cannot be explained here due to length restrictions. The proof relies on the already mentioned uniqueness of the proposed partitioning of spines, on which steps (1) to (8) are based.

References

[1] F. Jelinek and J.D. Lafferty. Computation of the probability of initial substring generation by stochastic context-free grammars. *Computational Linguistics*, 17(3):315–323, 1991.

[2] M.-J. Nederhof, A. Sarkar, and G. Satta. Prefix probabilities from stochastic tree adjoining grammars. In *36th Annual Meeting of the ACL, Proceedings of the Conference*, Montreal, Canada, August 1998. To appear.

[3] Y. Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics*, volume 2, pages 426–432, Nantes, August 1992.

[4] A. Stolcke. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):167–201, 1995.